Electric Water Heater

version 1.0

Generated by Doxygen 1.8.18

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 sTask_t Struct Reference

Data Fields

- void(∗ pTask)(void)
- uint16 Delay
- uint16 Period
- uint8 RunMe

3.1.1 Field Documentation

3.1.1.1 Delay

```
uint16 Delay
```

Delay (ticks) until the function will (next) be run

3.1.1.2 Period

```
uint16 Period
```

Interval (ticks) between subsequent runs.

3.1.1.3 pTask

```
void(* pTask) (void)
```

Pointer to the task (must be a 'void (void)' function)

3.1.1.4 RunMe

```
uint8 RunMe
```

Incremented (by scheduler) when task is due to execute

The documentation for this struct was generated from the following file:

- Scheduler.h

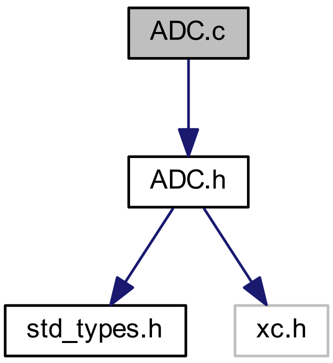
Chapter 4

File Documentation

4.1 ADC.c File Reference

ADC Module Source file for this program.

```
#include "ADC.h"
```
Include dependency graph for ADC.c:



Functions

- void ADC_Init (void)
- uint16 ADC_ReadChannel (uint8 channel)

4.1.1 Detailed Description

ADC Module Source file for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.1.2 Function Documentation

4.1.2.1 ADC_Init()

```
void ADC_Init (
             void  )
```

Brief: This is The ADC Module Initialization Function

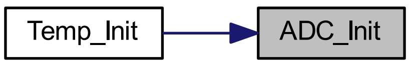
Parameters

void	

Returns

void

Here is the caller graph for this function:



4.1.2.2 ADC_ReadChannel()

```
uint16 ADC_ReadChannel (
            uint8 channel )
```

Brief: This is The ADC Module Read channel Function

Parameters

channel	unsigned char to select certain channel of ADC module

Returns

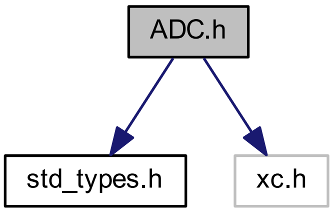
unsigned int to get the reading of ADC selected channel

4.2 ADC.h File Reference

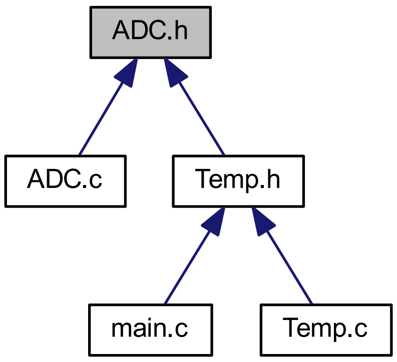
ADC Module header file for this program.

```
#include "std_types.h"
#include <xc.h>
```

Include dependency graph for ADC.h:



This graph shows which files directly or indirectly include this file:



Functions

- void ADC_Init (void)
- uint16 ADC_ReadChannel (uint8 channel)

4.2.1 Detailed Description

ADC Module header file for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.2.2 Function Documentation

4.2.2.1 ADC_Init()

```
void ADC_Init (
            void  )
```

Brief: This is The ADC Module Initialization Function

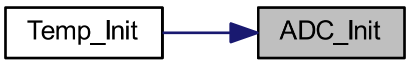
Parameters

void	

Returns

void

Here is the caller graph for this function:



4.2.2.2 ADC_ReadChannel()

```
uint16 ADC_ReadChannel (
            uint8 channel )
```

Brief: This is The ADC Module Read channel Function

Parameters

channel	unsigned char to select certain channel of ADC module

Returns

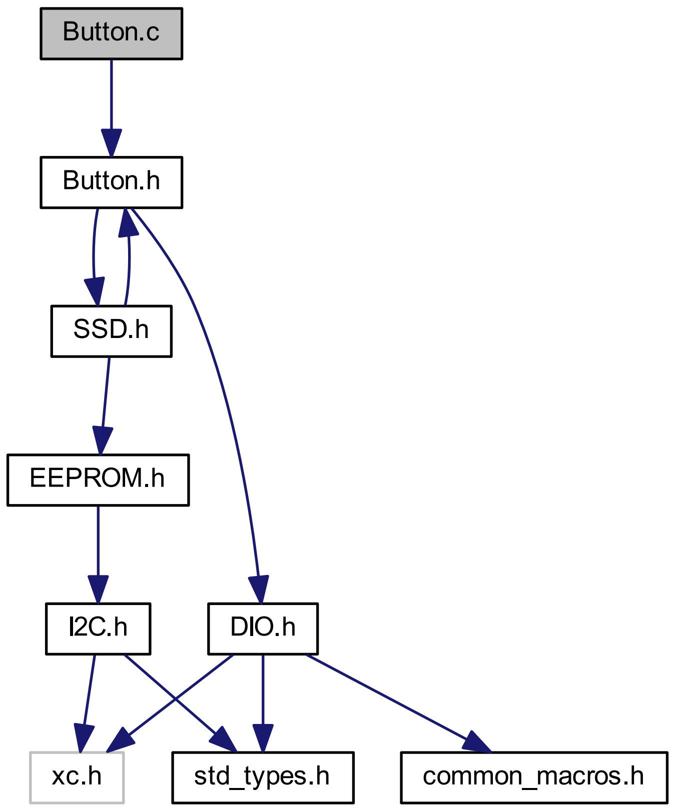
unsigned int to get the reading of ADC selected channel

4.3 Button.c File Reference

Button Source File for this program.

```
#include "Button.h"
```

Include dependency graph for Button.c:



Functions

- void Button_Init (void)
- void Button_Update (void)

Variables

- SSD_MODE_States_t **SSD_MODE_State**
- SW_ON_OFF_States_t **SW_ON_OFF_State** = OFF_WAIT
- System_States_t **System_State** = System_OFF
- uint8 **SW_UP_isPressed** =0
- uint8 **SW_DOWN_isPressed** =0

4.3.1 Detailed Description

Button Source File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.3.2 Function Documentation

4.3.2.1 Button_Init()

```
void Button_Init (
             void  )
```

Brief: This is the Button initialization function to initialize ON/OFF button , Up button ,Down button

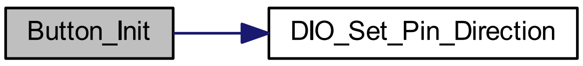
Parameters

void	

Returns

void

Here is the call graph for this function:



4.3.2.2 Button_Update()

```
void Button_Update (
             void  )
```

Brief: This is the Button update Task to update system state by check ON/OFF button state and check Up button or Down button state if anyone is pressed change state of SSD form normal mode to setting mode

Parameters

void	

Returns

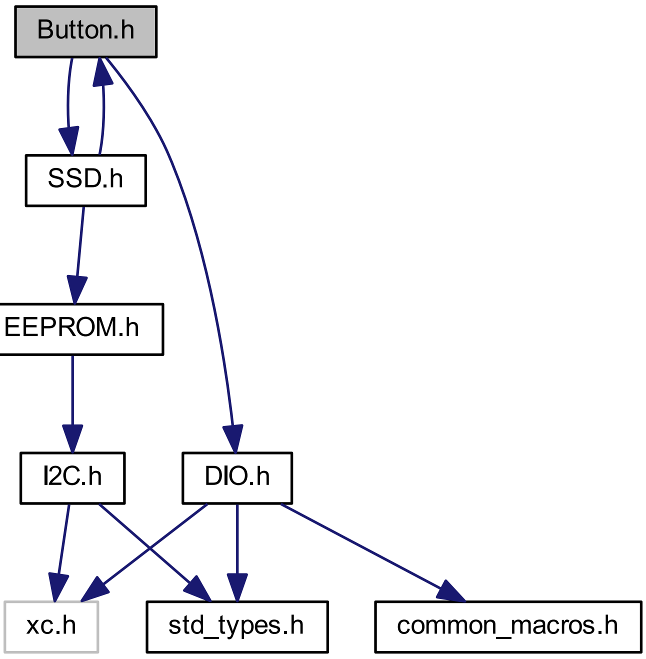
void

4.4 Button.h File Reference

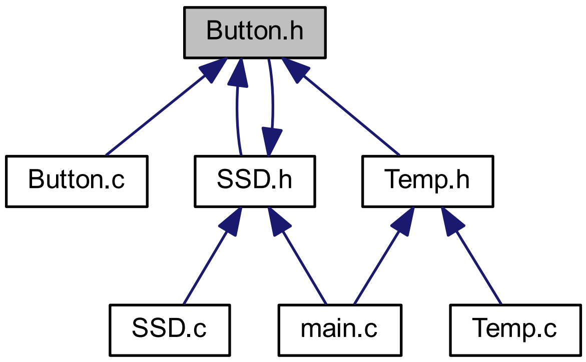
Button Header File for this program.

```
#include "SSD.h"
#include "DIO.h"
```

Include dependency graph for Button.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define SW_ON_OFF_PIN 1
- #define SW_UP_PIN 2
- #define SW_DOWN_PIN 0
- #define SW_PORT B
- #define IS_PRESSED 0x00
- #define IS_RELEASED 0x01

Enumerations

- enum **System_States_t** { **System_ON**, **System_OFF** }
- enum **SW_ON_OFF_States_t** { **ON**, **ON_WAIT**, **OFF**, **OFF_WAIT** }

Functions

- void Button_Init (void)
- void Button_Update (void)

4.4.1 Detailed Description

Button Header File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.4.2 Macro Definition Documentation

4.4.2.1 IS_PRESSED

```
#define IS_PRESSED 0x00
```

button is pressed detector

4.4.2.2 IS_RELEASED

```
#define IS_RELEASED 0x01
```

button is released detector

4.4.2.3 SW_DOWN_PIN

```
#define SW_DOWN_PIN 0
```

Down Button Pin

4.4.2.4 SW_ON_OFF_PIN

```
#define SW_ON_OFF_PIN 1
```

ON/OFF Button Pin

4.4.2.5 SW_PORT

```
#define SW_PORT B
```

Buttons Port

4.4.2.6 SW_UP_PIN

```
#define SW_UP_PIN 2
```

Up Button Pin

4.4.3 Function Documentation

4.4.3.1 Button_Init()

```
void Button_Init (
             void  )
```

Brief: This is the Button initialization function to initialize ON/OFF button , Up button ,Down button

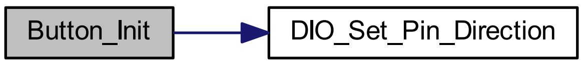
Parameters

void	

Returns

void

Here is the call graph for this function:



4.4.3.2 Button_Update()

```
void Button_Update (
             void  )
```

Brief: This is the Button update Task to update system state by check ON/OFF button state and check Up button or Down button state if anyone is pressed change state of SSD form normal mode to setting mode

Parameters

void	

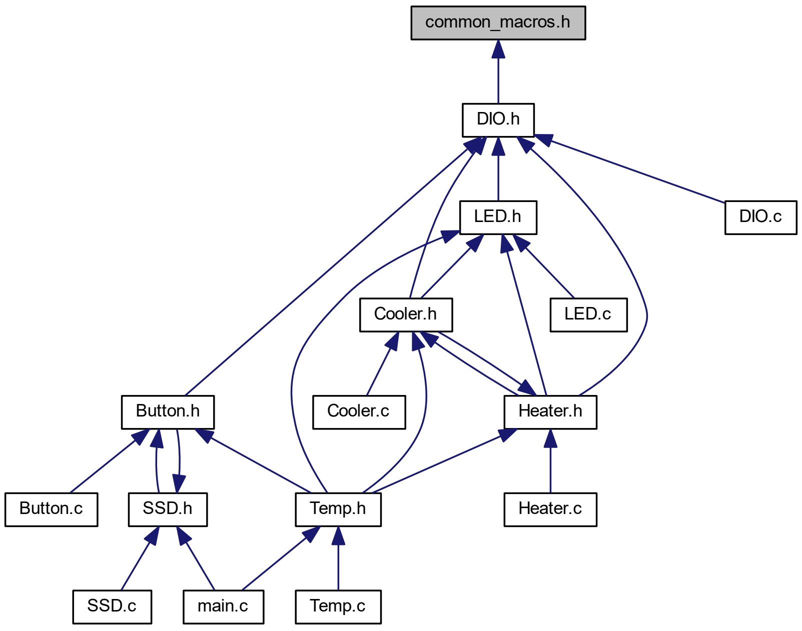
Returns

void

4.5 common_macros.h File Reference

Common Macros header file for this program.

This graph shows which files directly or indirectly include this file:



Macros

- #define SET_BIT(REG, BIT) (REG|=(1<<BIT))
- #define CLEAR_BIT(REG, BIT) (REG&=(∼(1<<BIT)))
- #define GET_BIT(REG, BIT) ((REG>>BIT)&1)
- #define TOGGLE_BIT(REG, BIT) (REG^=(1<<BIT))

4.5.1 Detailed Description

Common Macros header file for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.5.2 Macro Definition Documentation

4.5.2.1 CLEAR_BIT

```
#define CLEAR_BIT(
              REG,
              BIT ) (REG&=(~(1<<BIT)))
```

Clear a certain bit in any register

4.5.2.2 GET_BIT

```
#define GET_BIT(
              REG,
              BIT ) ((REG>>BIT)&1)
```

Clear a certain bit in any register

4.5.2.3 SET_BIT

```
#define SET_BIT(
              REG,
              BIT ) (REG|=(1<<BIT))
```

Set a certain bit in any register

4.5.2.4 TOGGLE_BIT

```
#define TOGGLE_BIT(
              REG,
              BIT ) (REG^=(1<<BIT))
```

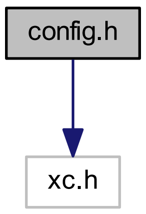
Toggle a certain bit in any register

4.6 config.h File Reference

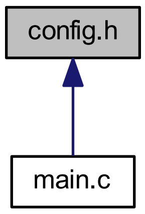
PIC16F877A Configuration Bit Settings file for this program.

```
#include <xc.h>
```

Include dependency graph for config.h:



This graph shows which files directly or indirectly include this file:



4.6.1 Detailed Description

PIC16F877A Configuration Bit Settings file for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

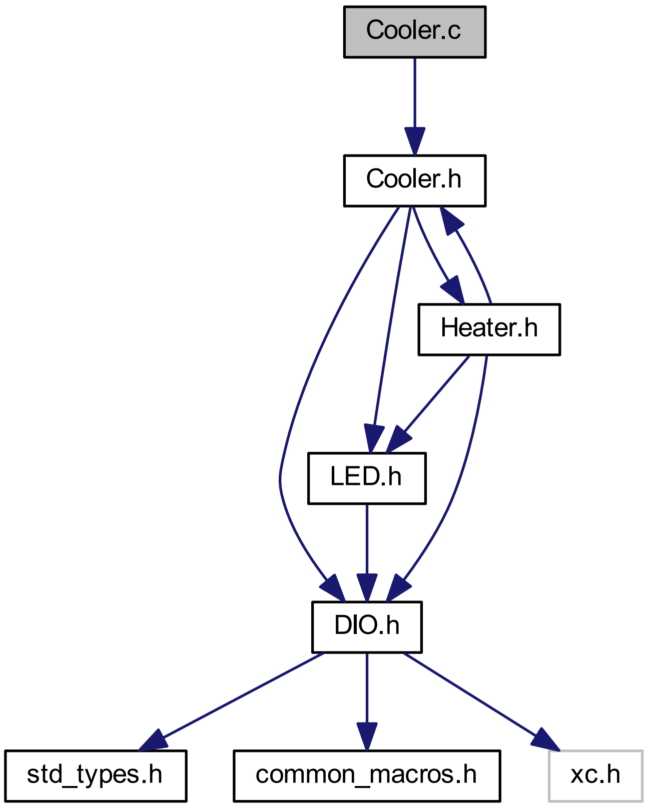
1.0

4.7 Cooler.c File Reference

Cooler Element Source File for this program.

```
#include "Cooler.h"
```

Include dependency graph for Cooler.c:



Functions

- void Cooler_OFF (void)
- void Cooler_Init (void)
- void Cooler_ON (void)
- void Cooler_Update (void)

4.7.1 Detailed Description

Cooler Element Source File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.7.2 Function Documentation

4.7.2.1 Cooler_Init()

```
void Cooler_Init (
             void   )
```

Brief: This is function to Set Cooler port direction as output with OFF State

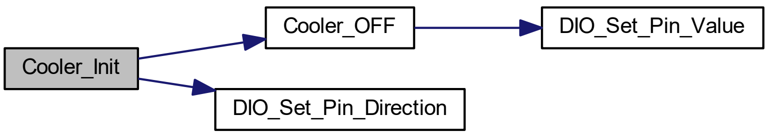
Parameters

void	

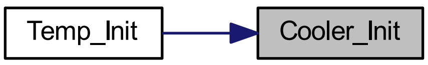
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.2.2 Cooler_OFF()

```
void Cooler_OFF (
             void   )
```

Brief: This is function to Turn Cooler OFF

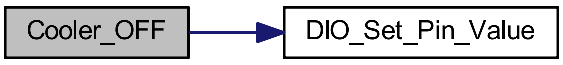
Parameters

void	

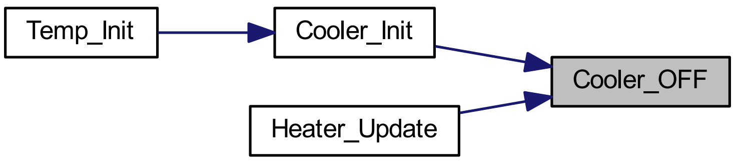
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.2.3 Cooler_ON()

```
void Cooler_ON (
             void  )
```

Brief: This is function to Turn Cooler ON

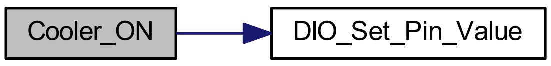
Parameters

void	

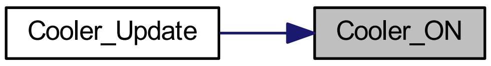
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.2.4 Cooler_Update()

```
void Cooler_Update (
            void )
```

Brief: This is function to Turn Cooler ON and Turn Heater OFF and Turn LED OFF

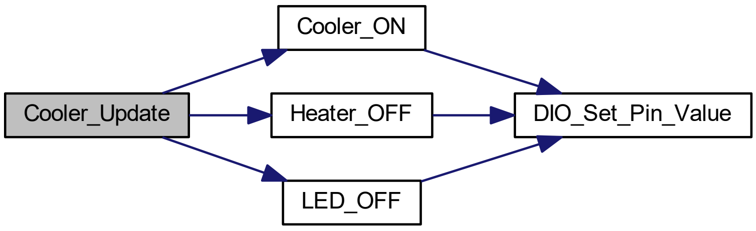
Parameters

void	

Returns

void

Here is the call graph for this function:

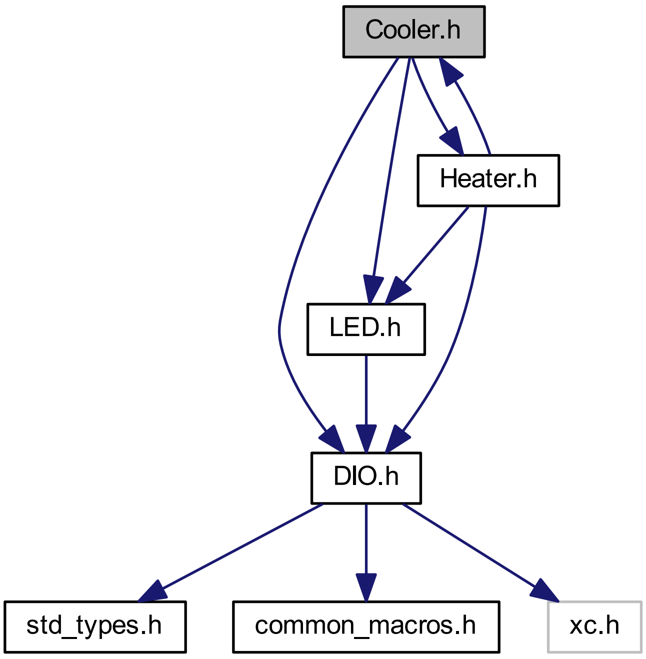


4.8 Cooler.h File Reference

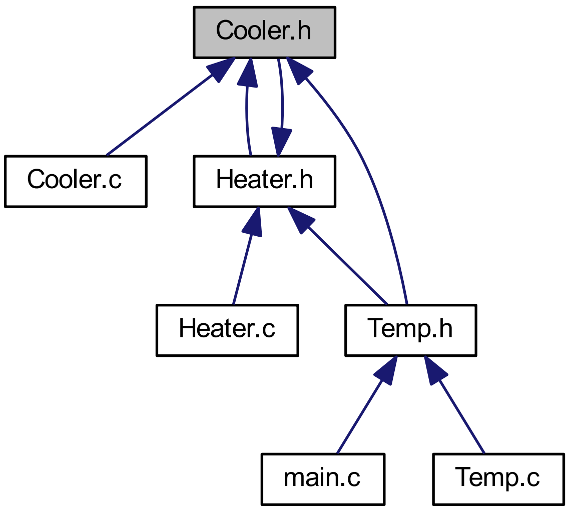
Cooler Element Header file for this program.

```
#include "DIO.h"
#include "LED.h"
#include "Heater.h"
```

Include dependency graph for Cooler.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define COOLER_PIN 2
- #define COOLER_PORT C

Functions

- void Cooler_OFF (void)
- void Cooler_Init (void)
- void Cooler_ON (void)
- void Cooler_Update (void)

4.8.1 Detailed Description

Cooler Element Header file for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.8.2 Macro Definition Documentation

4.8.2.1 COOLER_PIN

```
#define COOLER_PIN 2
```

Cooler Pin

4.8.2.2 COOLER_PORT

```
#define COOLER_PORT C
```

Cooler Port

4.8.3 Function Documentation

4.8.3.1 Cooler_Init()

```
void Cooler_Init (
             void  )
```

Brief: This is function to Set Cooler port direction as output with OFF State

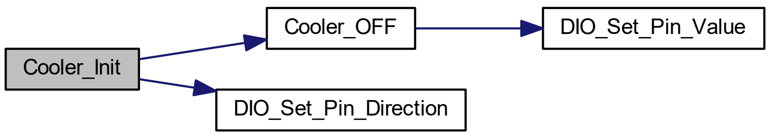
Parameters

void	

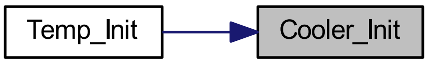
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.8.3.2 Cooler_OFF()

```
void Cooler_OFF (
             void  )
```

Brief: This is function to Turn Cooler OFF

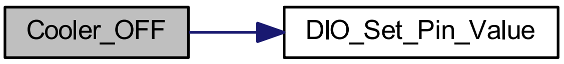
Parameters

void	

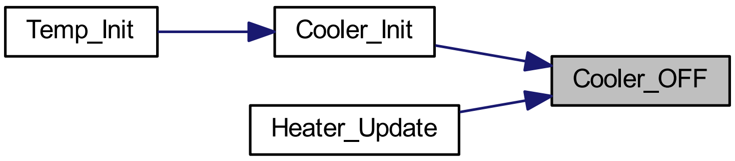
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.8.3.3 Cooler_ON()

```
void Cooler_ON (
             void  )
```

Brief: This is function to Turn Cooler ON

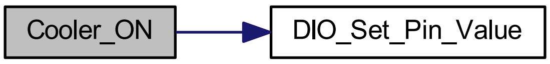
Parameters

void	

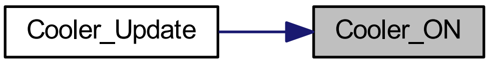
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.8.3.4 Cooler_Update()

```
void Cooler_Update (
             void  )
```

Brief: This is function to Turn Cooler ON and Turn Heater OFF and Turn LED OFF

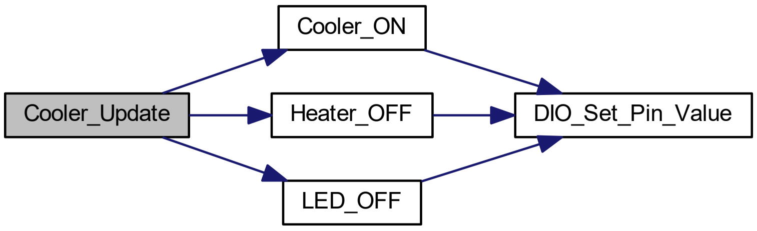
Parameters

void	

Returns

void

Here is the call graph for this function:

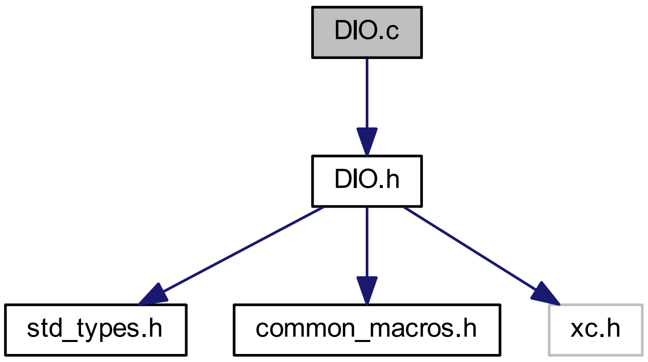


4.9 DIO.c File Reference

DIO Module Source File for this program.

```
#include "DIO.h"
```

Include dependency graph for DIO.c:



Functions

- void DIO_Set_Port_Direction (uint8 portNumber, uint8 direction)
- void DIO_Set_Port_Value (uint8 portNumber, uint8 value)
- uint8 DIO_Read_Port_Value (uint8 portNumber)
- uint8 DIO_Read_Pin_Value (uint8 portNumber, uint8 index)
- void DIO_Set_Pin_Value (uint8 portNumber, uint8 index, uint8 value)
- void DIO_Set_Pin_Direction (uint8 portNumber, uint8 index, uint8 direction)

4.9.1 Detailed Description

DIO Module Source File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.9.2 Function Documentation

4.9.2.1 DIO_Read_Pin_Value()

```
uint8 DIO_Read_Pin_Value (
            uint8 portNumber,
            uint8 index )
```

Brief: This is function to select certain pin of pins (0->7) of port of ports (A,B,C,D,E) to get the value

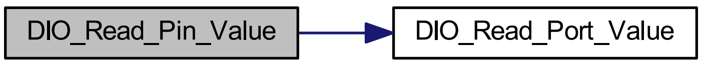
Parameters

portNumber	unsigned char portNumber to select port
index	unsigned char index to select pin

Returns

unsigned char to get value of pin

Here is the call graph for this function:



4.9.2.2 DIO_Read_Port_Value()

```
uint8 DIO_Read_Port_Value (
            uint8 portNumber )
```

Brief: This is function to select certain port of ports (A,B,C,D,E) to get the value

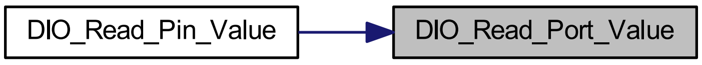
Parameters

portNumber	unsigned char portNumber to select port

Returns

unsigned char to get value of port

Here is the caller graph for this function:



4.9.2.3 DIO_Set_Pin_Direction()

```
void DIO_Set_Pin_Direction (
             uint8 portNumber,
             uint8 index,
             uint8 direction )
```

Brief: This is function to set certain pin of pins (0->7) of port of ports (A,B,C,D,E) direction (OUTPUT , INPUT)

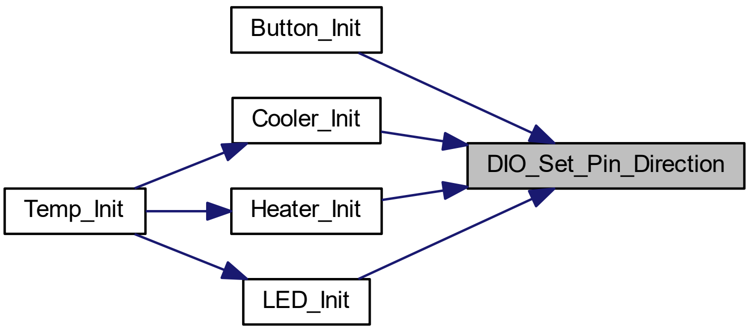
Parameters

portNumber	unsigned char portNumber to select port
index	unsigned char index to select pin
direction	unsigned char direction to select direction of pin

Returns

void

Here is the caller graph for this function:



4.9.2.4 DIO_Set_Pin_Value()

```
void DIO_Set_Pin_Value (
             uint8 portNumber,
             uint8 index,
             uint8 value )
```

Brief: This is function to select certain pin of pins (0->7) of port of ports (A,B,C,D,E) to set the value (LOW , HIGH , any value)

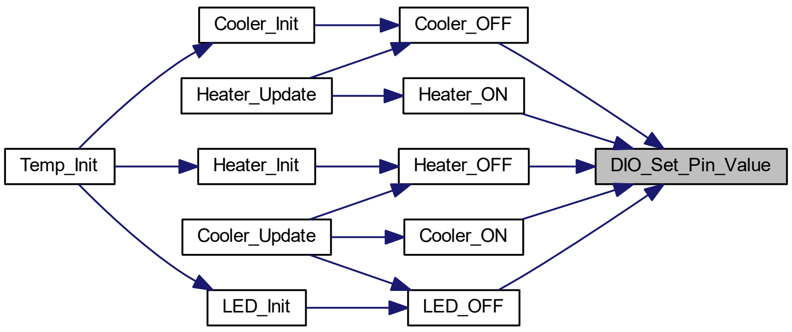
Parameters

portNumber	unsigned char portNumber to select port
index	unsigned char index to select pin
value	unsigned char value to value

Returns

void

Here is the caller graph for this function:



4.9.2.5 DIO_Set_Port_Direction()

```
void DIO_Set_Port_Direction (
             uint8 portNumber,
             uint8 direction )
```

Brief: This is function to set certain port of ports (A,B,C,D,E) direction (OUTPUT , INPUT)

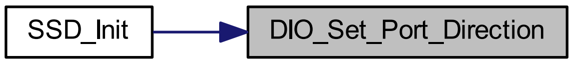
Parameters

portNumber	unsigned char portNumber to select port
direction	unsigned char direction to select direction of port

Returns

void

Here is the caller graph for this function:



4.9.2.6 DIO_Set_Port_Value()

```
void DIO_Set_Port_Value (
             uint8 portNumber,
             uint8 value )
```

Brief: This is function to set certain port of ports (A,B,C,D,E) value (LOW , HIGH , any value)

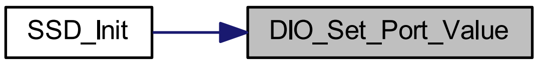
Parameters

portNumber	unsigned char portNumber to select port
value	unsigned char direction to set value

Returns

void

Here is the caller graph for this function:

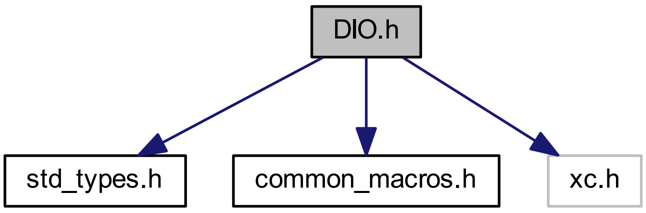


4.10 DIO.h File Reference

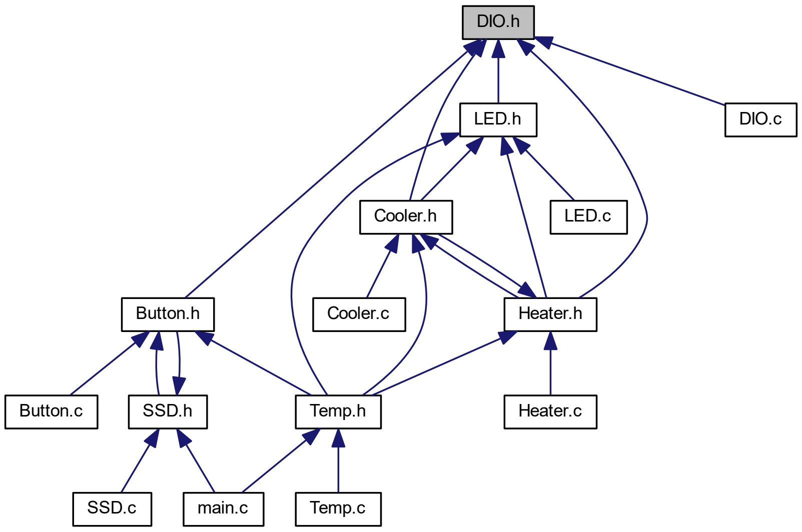
DIO Module Header File for this program.

```
#include "std_types.h"
#include "common_macros.h"
#include <xc.h>
```

Include dependency graph for DIO.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define OUTPUT_PORT 0x00
- #define INPUT_PORT 0xFF
- #define OUTPUT_PIN 0x00
- #define INPUT_PIN 0x01
- #define HIGH_PORT 0xFF
- #define LOW_PORT 0x00
- #define HIGH_PIN 0x01
- #define LOW_PIN 0x00

Enumerations

- enum **PORTS_t** {
  A, **B**, **C**, **D**,
  E }

Functions

- void DIO_Set_Port_Direction (uint8 portNumber, uint8 direction)
- void DIO_Set_Port_Value (uint8 portNumber, uint8 value)
- uint8 DIO_Read_Port_Value (uint8 portNumber)
- uint8 DIO_Read_Pin_Value (uint8 portNumber, uint8 index)
- void DIO_Set_Pin_Value (uint8 portNumber, uint8 index, uint8 value)
- void DIO_Set_Pin_Direction (uint8 portNumber, uint8 index, uint8 direction)

4.10.1 Detailed Description

DIO Module Header File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.10.2 Macro Definition Documentation

4.10.2.1 HIGH_PIN

```
#define HIGH_PIN 0x01
```

Pin is High Voltage

4.10.2.2 HIGH_PORT

```
#define HIGH_PORT 0xFF
```

Port is High Voltage

4.10.2.3 INPUT_PIN

```
#define INPUT_PIN 0x01
```

Pin is Input Direction

4.10.2.4 INPUT_PORT

```
#define INPUT_PORT 0xFF
```

Port is Input Direction

4.10.2.5 LOW_PIN

```
#define LOW_PIN 0x00
```

Pin is Low Voltage

4.10.2.6 LOW_PORT

```
#define LOW_PORT 0x00
```

Port is Low Voltage

4.10.2.7 OUTPUT_PIN

```
#define OUTPUT_PIN 0x00
```

Pin is Output Direction

4.10.2.8 OUTPUT_PORT

```
#define OUTPUT_PORT 0x00
```

Port is Output Direction

4.10.3 Function Documentation

4.10.3.1 DIO_Read_Pin_Value()

```
uint8 DIO_Read_Pin_Value (
            uint8 portNumber,
            uint8 index )
```

Brief: This is function to select certain pin of pins (0->7) of port of ports (A,B,C,D,E) to get the value

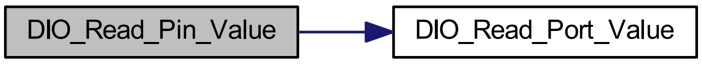
Parameters

portNumber	unsigned char portNumber to select port
index	unsigned char index to select pin

Returns

unsigned char to get value of pin

Here is the call graph for this function:



4.10.3.2 DIO_Read_Port_Value()

```
uint8 DIO_Read_Port_Value (
            uint8 portNumber )
```

Brief: This is function to select certain port of ports (A,B,C,D,E) to get the value

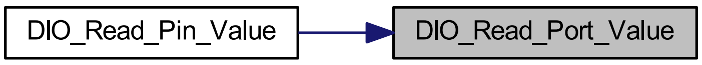
Parameters

portNumber	unsigned char portNumber to select port

Returns

unsigned char to get value of port

Here is the caller graph for this function:



4.10.3.3 DIO_Set_Pin_Direction()

```
void DIO_Set_Pin_Direction (
             uint8 portNumber,
             uint8 index,
             uint8 direction )
```

Brief: This is function to set certain pin of pins (0->7) of port of ports (A,B,C,D,E) direction (OUTPUT , INPUT)

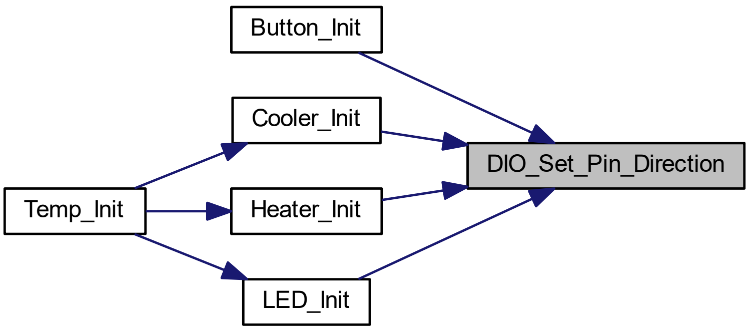
Parameters

portNumber	unsigned char portNumber to select port
index	unsigned char index to select pin
direction	unsigned char direction to select direction of pin

Returns

void

Here is the caller graph for this function:



4.10.3.4 DIO_Set_Pin_Value()

```
void DIO_Set_Pin_Value (
             uint8 portNumber,
             uint8 index,
             uint8 value )
```

Brief: This is function to select certain pin of pins (0->7) of port of ports (A,B,C,D,E) to set the value (LOW , HIGH , any value)

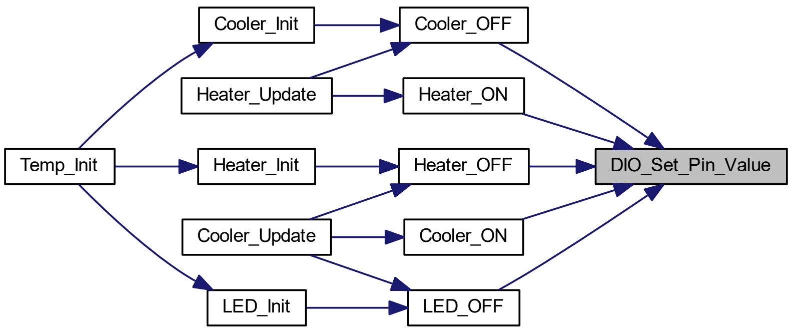
Parameters

portNumber	unsigned char portNumber to select port
index	unsigned char index to select pin
value	unsigned char value to value

Returns

void

Here is the caller graph for this function:



4.10.3.5 DIO_Set_Port_Direction()

```
void DIO_Set_Port_Direction (
             uint8 portNumber,
             uint8 direction )
```

Brief: This is function to set certain port of ports (A,B,C,D,E) direction (OUTPUT , INPUT)

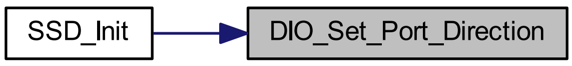
Parameters

portNumber	unsigned char portNumber to select port
direction	unsigned char direction to select direction of port

Returns

void

Here is the caller graph for this function:



4.10.3.6 DIO_Set_Port_Value()

```
void DIO_Set_Port_Value (
             uint8 portNumber,
             uint8 value )
```

Brief: This is function to set certain port of ports (A,B,C,D,E) value (LOW , HIGH , any value)

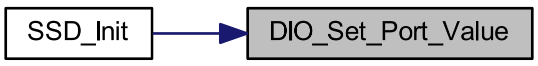
Parameters

portNumber	unsigned char portNumber to select port
value	unsigned char direction to set value

Returns

void

Here is the caller graph for this function:

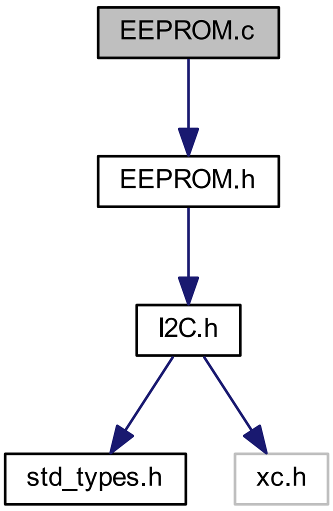


4.11 EEPROM.c File Reference

EEPROM Module Source File for this program.

`#include "EEPROM.h"`

Include dependency graph for EEPROM.c:



Functions

- void EEPROM_Init (void)
- void EEPROM_Write (uint8 address, uint8 data)
- uint8 EEPROM_Read (uint8 address)

4.11.1 Detailed Description

EEPROM Module Source File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.11.2 Function Documentation

4.11.2.1 EEPROM_Init()

```
void EEPROM_Init (
             void  )
```

Brief: This is the Initialization of EEPROM function to Intialize ECU as Maseter

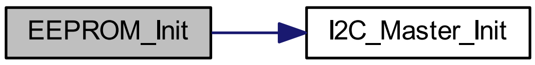
Parameters

void	

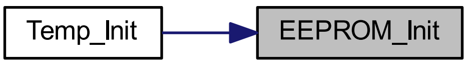
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.2.2 EEPROM_Read()

```
uint8 EEPROM_Read (
            uint8 address )
```

Brief: This is the External EEPROM Write function to write a certain data at certain address of external EEPROM

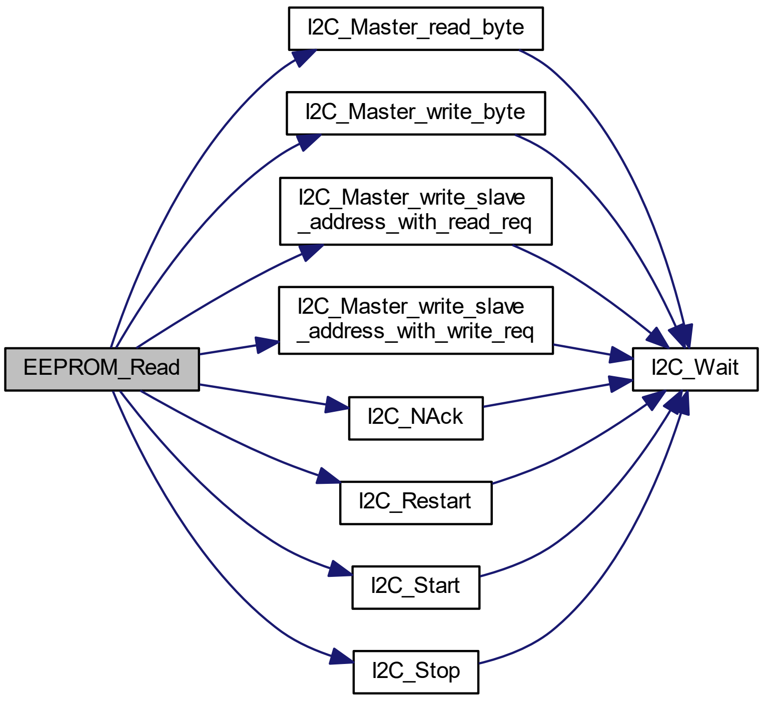
Parameters

address	unsigned char address to select a certain address that you want to read from it

Returns

unsigned char data at this certain address

Here is the call graph for this function:



4.11.2.3 EEPROM_Write()

```
void EEPROM_Write (
             uint8 address,
             uint8 data )
```

Brief: This is the External EEPROM Write function to write a certain data at certain address of external EEPROM

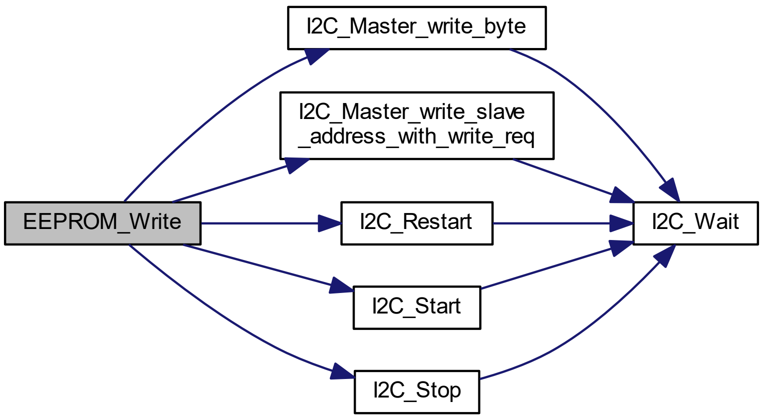
Parameters

address	unsigned char address to select a certain address that you want to write at it
data	unsigned char data to write data at certain address

Returns

void

Here is the call graph for this function:

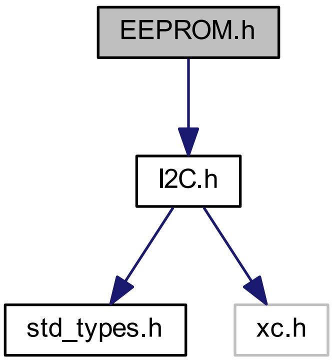


4.12 EEPROM.h File Reference

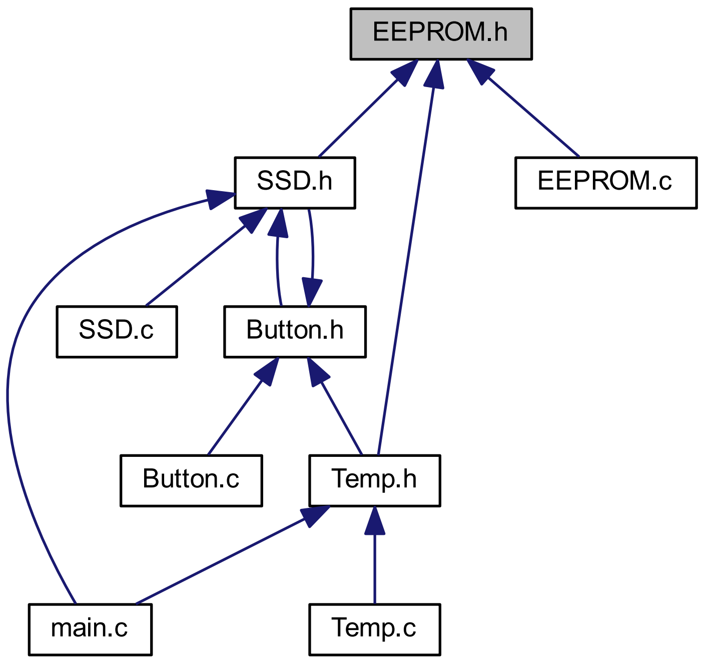
EEPROM Module header file for this program.

```
#include "I2C.h"
```

Include dependency graph for EEPROM.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define EEPROM_ADDRESS 0x50

Functions

- void EEPROM_Init (void)
- void EEPROM_Write (uint8 address, uint8 data)
- uint8 EEPROM_Read (uint8 address)

4.12.1 Detailed Description

EEPROM Module header file for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.12.2 Macro Definition Documentation

4.12.2.1 EEPROM_ADDRESS

```
#define EEPROM_ADDRESS 0x50
```

EEPROM Address

4.12.3 Function Documentation

4.12.3.1 EEPROM_Init()

```
void EEPROM_Init (
            void  )
```

Brief: This is the Initialization of EEPROM function to Intialize ECU as Maseter

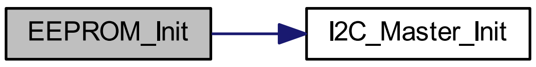
Parameters

void	

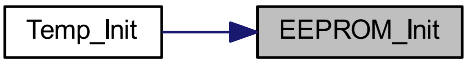
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.3.2 EEPROM_Read()

```
uint8 EEPROM_Read (
            uint8 address )
```

Brief: This is the External EEPROM Write function to write a certain data at certain address of external EEPROM

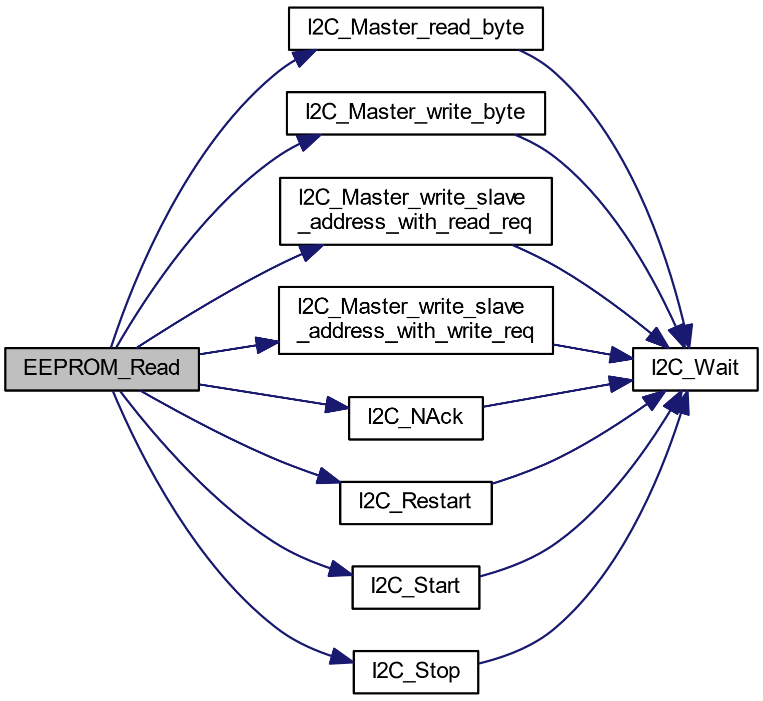
Parameters

address	unsigned char address to select a certain address that you want to read from it

Returns

unsigned char data at this certain address

Here is the call graph for this function:



4.12.3.3 EEPROM_Write()

```
void EEPROM_Write (
            uint8 address,
            uint8 data )
```

Brief: This is the External EEPROM Write function to write a certain data at certain address of external EEPROM

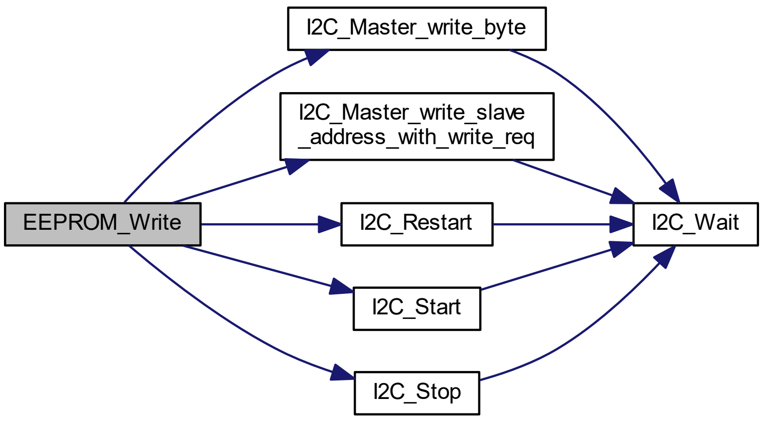
Parameters

address	unsigned char address to select a certain address that you want to write at it
data	unsigned char data to write data at certain address

Returns

void

Here is the call graph for this function:

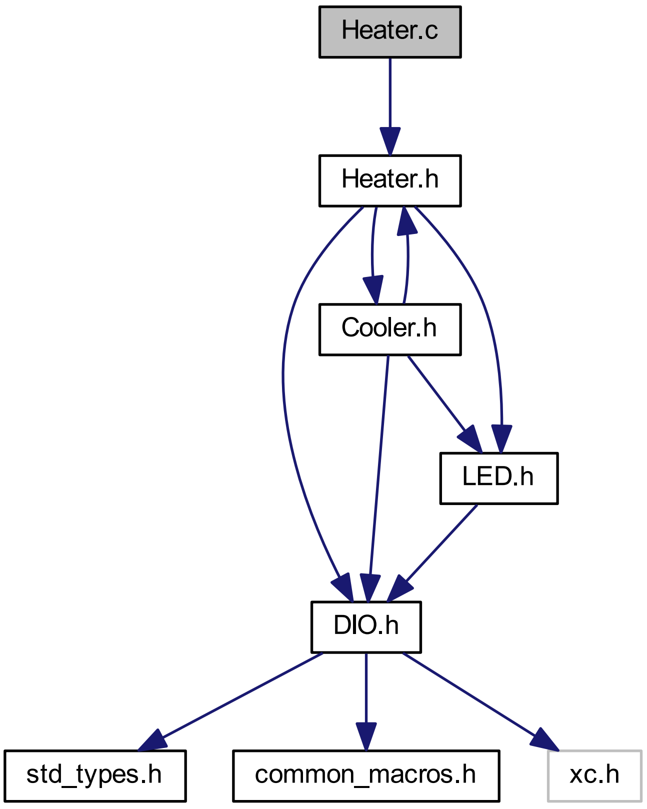


4.13 Heater.c File Reference

Heater element Source file for this program.

```
#include "Heater.h"
```

Include dependency graph for Heater.c:



Functions

- void Heater_OFF (void)
- void Heater_Init (void)
- void Heater_ON (void)
- void Heater_Update (void)

4.13.1 Detailed Description

Heater element Source file for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.13.2 Function Documentation

4.13.2.1 Heater_Init()

```
void Heater_Init (
             void  )
```

Brief: This is function to Set Heater port direction as output with OFF State

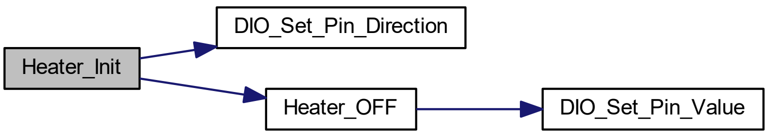
Parameters

void	

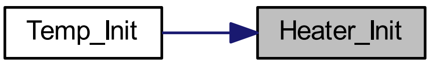
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.13.2.2 Heater_OFF()

```
void Heater_OFF (
             void  )
```

Brief: This is function to Turn Heater OFF

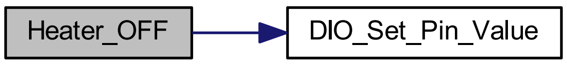
Parameters

void	

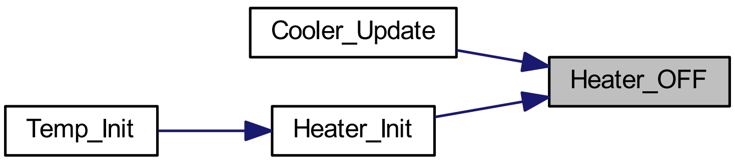
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.13.2.3 Heater_ON()

```
void Heater_ON (
            void  )
```

Brief: This is function to Turn Cooler ON

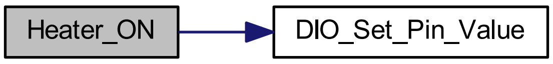
Parameters

void	

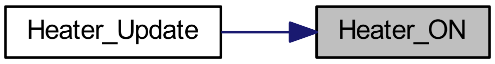
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.13.2.4 Heater_Update()

```
void Heater_Update (
            void  )
```

Brief: This is function to Turn Heater ON and Turn Cooler OFF and Turn LED ON

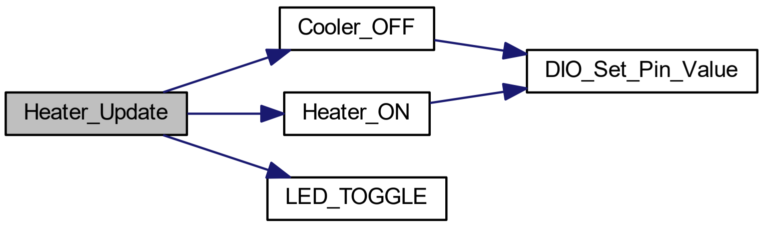
Parameters

void	

Returns

void

Here is the call graph for this function:

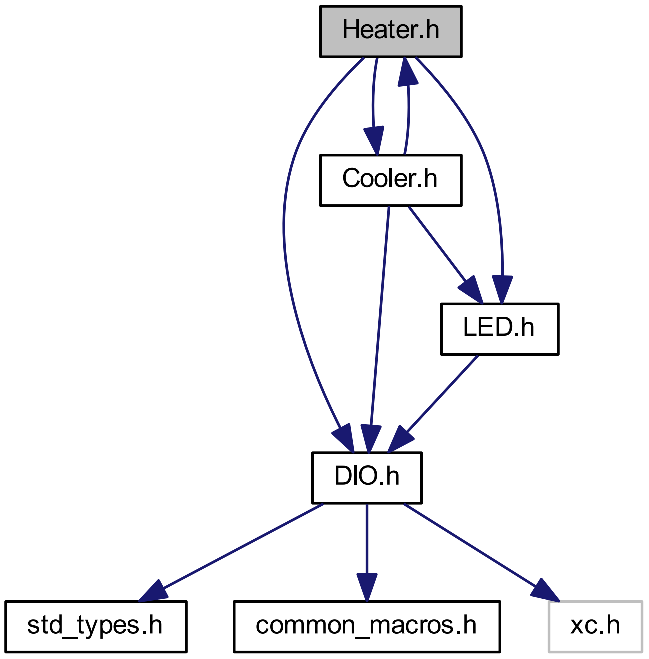


4.14 Heater.h File Reference

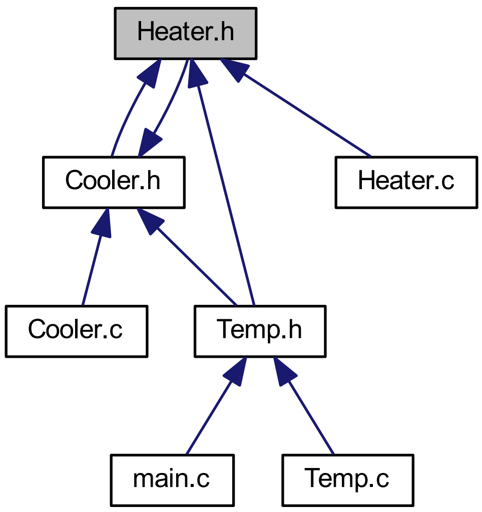
Heater Element Header File for this program.

```
#include "DIO.h"
#include "Cooler.h"
#include "LED.h"
```

Include dependency graph for Heater.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define HEATER_PIN 5
- #define HEATER_PORT C

Functions

- void Heater_OFF (void)
- void Heater_Init (void)
- void Heater_ON (void)
- void Heater_Update (void)

4.14.1 Detailed Description

Heater Element Header File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.14.2 Macro Definition Documentation

4.14.2.1 HEATER_PIN

```
#define HEATER_PIN 5
```

Heater Pin

4.14.2.2 HEATER_PORT

```
#define HEATER_PORT C
```

Heater Port

4.14.3 Function Documentation

4.14.3.1 Heater_Init()

```
void Heater_Init (
             void  )
```

Brief: This is function to Set Heater port direction as output with OFF State

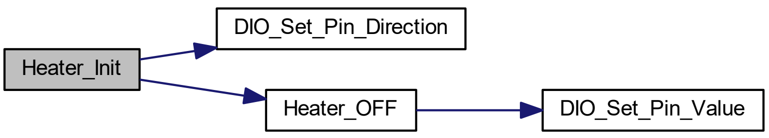
Parameters

void	

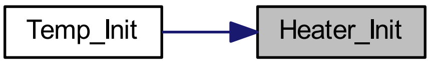
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.14.3.2 Heater_OFF()

```
void Heater_OFF (
             void  )
```

Brief: This is function to Turn Heater OFF

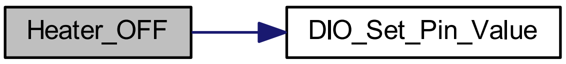
Parameters

void	

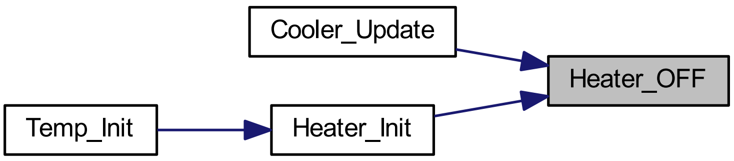
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.14.3.3 Heater_ON()

```
void Heater_ON (
             void  )
```

Brief: This is function to Turn Cooler ON

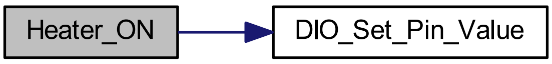
Parameters

void	

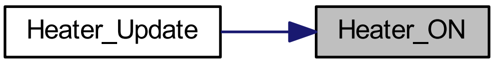
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.14.3.4 Heater_Update()

```
void Heater_Update (
             void  )
```

Brief: This is function to Turn Heater ON and Turn Cooler OFF and Turn LED ON

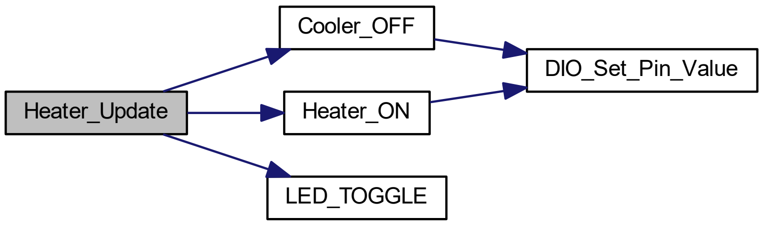
Parameters

void	

Returns

void

Here is the call graph for this function:

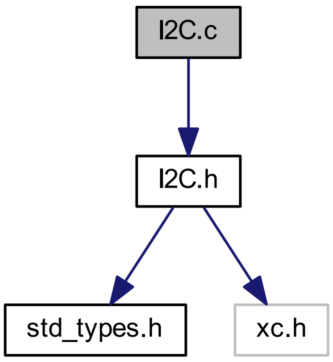


4.15 I2C.c File Reference

I2C Module Source File for this program.

```
#include "I2C.h"
```

Include dependency graph for I2C.c:



Functions

- void I2C_Master_Init (void)
- void I2C_Start (void)
- void I2C_Stop (void)
- void I2C_Restart (void)
- void I2C_Wait (void)
- void I2C_NAck (void)
- uint8 I2C_Master_write_slave_address_with_write_req (uint8 address)
- uint8 I2C_Master_write_slave_address_with_read_req (uint8 address)
- uint8 I2C_Master_write_byte (uint8 data)
- uint8 I2C_Master_read_byte (void)

4.15.1 Detailed Description

I2C Module Source File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.15.2 Function Documentation

4.15.2.1 I2C_Master_Init()

```
void I2C_Master_Init (
             void   )
```

Brief: This is the function to initialize ECU as Master Mode

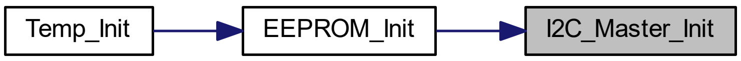
Parameters

void	

Returns

void

Here is the caller graph for this function:



4.15.2.2 I2C_Master_read_byte()

```
uint8 I2C_Master_read_byte (
            void  )
```

Brief: This is function to Master read data byte

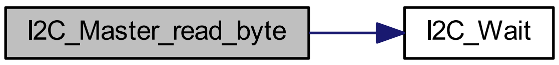
Parameters

void	

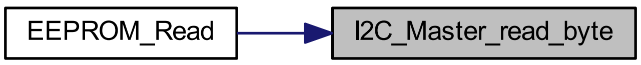
Returns

unsigned char data

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.2.3 I2C_Master_write_byte()

```
uint8 I2C_Master_write_byte (
            uint8 data )
```

Brief: This is function to Master write data byte

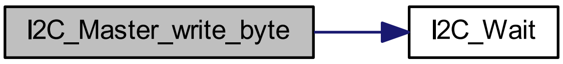
Parameters

data	unsigned char to write it

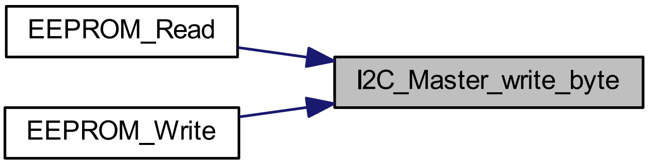
Returns

unsigned char

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.2.4 I2C_Master_write_slave_address_with_read_req()

```
uint8 I2C_Master_write_slave_address_with_read_req (
            uint8 address )
```

Brief: This is function to Master write address byte with read request

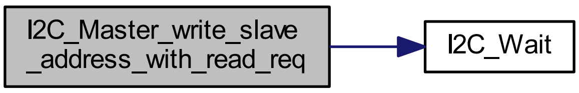
Parameters

address	unsigned char address to select a certain address that you want to read at it

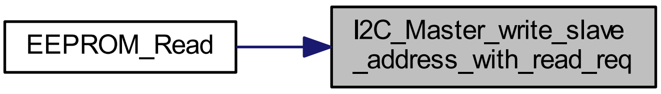
Returns

unsigned char true when finished

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.2.5 I2C_Master_write_slave_address_with_write_req()

```
uint8 I2C_Master_write_slave_address_with_write_req (
            uint8 address )
```

Brief: This is function to Master write address byte with write request

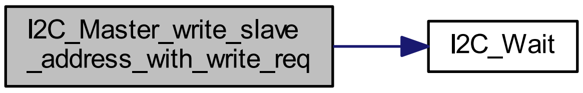
Parameters

address	unsigned char address to select a certain address that you want to write at it

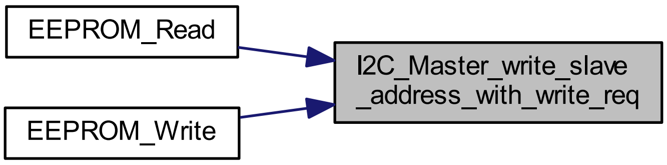
Returns

unsigned char true when finished

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.2.6 I2C_NAck()

```
void I2C_NAck (
             void  )
```

Brief: This is the I2C not Ack function

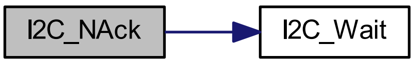
Parameters

void	

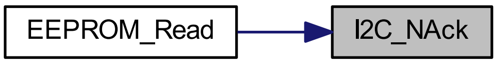
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.2.7 I2C_Restart()

```
void I2C_Restart (
             void  )
```

Brief: This is the function to Restart I2C communication protocol

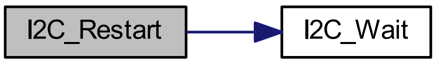
Parameters

void	

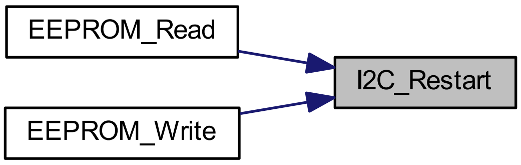
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.2.8 I2C_Start()

```
void I2C_Start (
             void  )
```

Brief: This is the function to Start I2C communication protocol

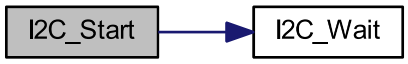
Parameters

void	

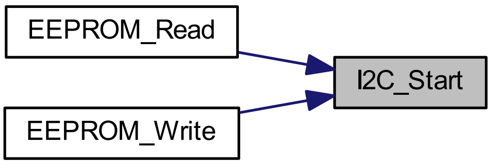
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.2.9 I2C_Stop()

```
void I2C_Stop (
             void  )
```

Brief: This is the function to Stop I2C communication protocol

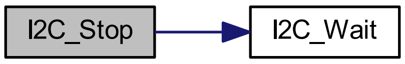
Parameters

void	

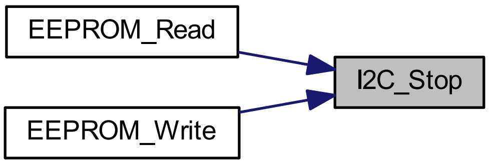
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.2.10 I2C_Wait()

```
void I2C_Wait (
                void  )
```

Brief: This is the I2C wait function

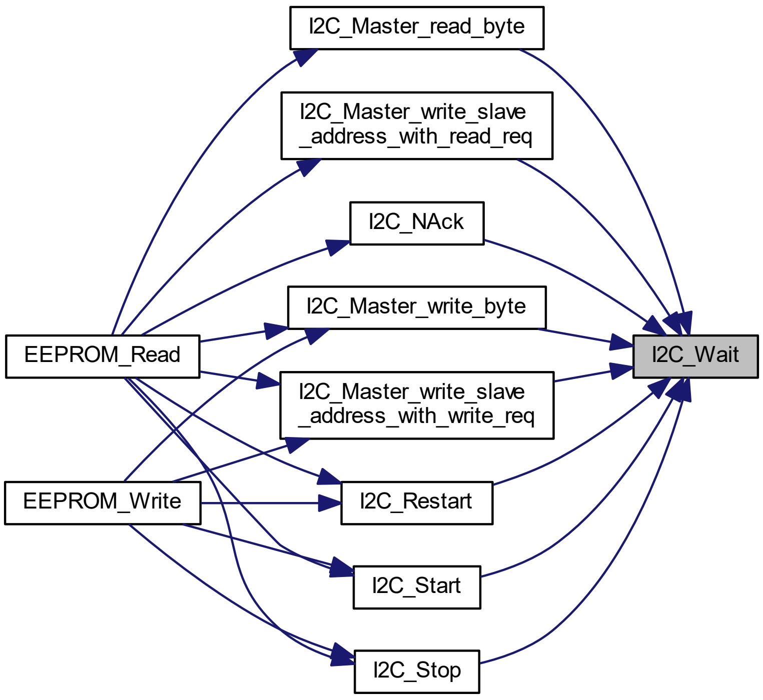
Parameters

void	

Returns

void

Here is the caller graph for this function:

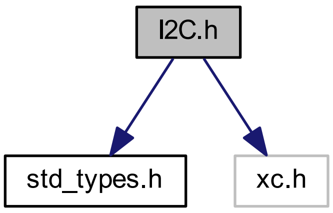


4.16 I2C.h File Reference

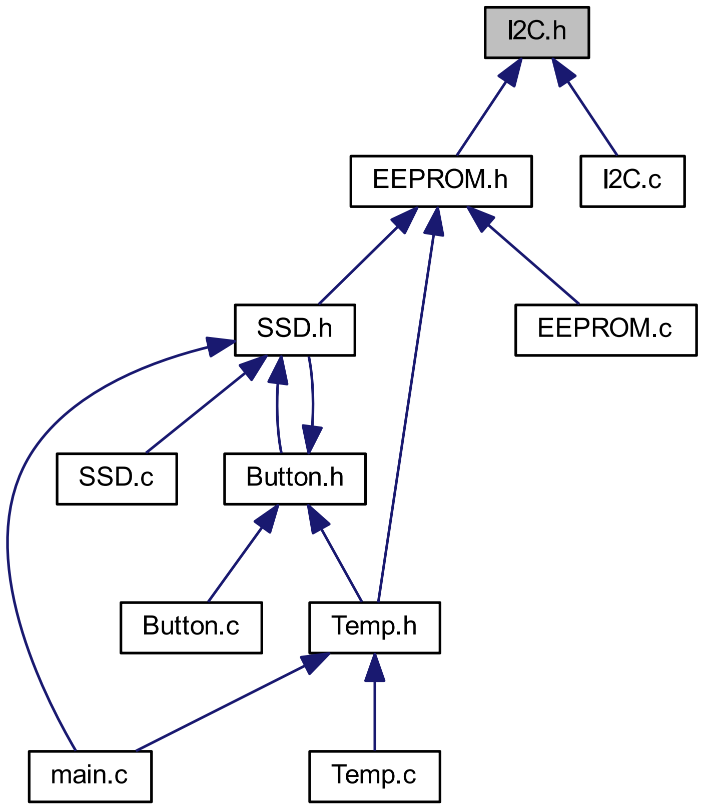
I2C Module Header File for this program.

```
#include "std_types.h"
#include <xc.h>
```

Include dependency graph for I2C.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define SCL_PIN 3
- #define SDA_PIN 4
- #define _XTAL_FREQ 8000000
- #define I2C_BAUDRATE 9600

Functions

- void I2C_Master_Init (void)
- void I2C_Start (void)
- void I2C_Stop (void)
- void I2C_Restart (void)
- void I2C_Wait (void)
- void I2C_NAck (void)
- uint8 I2C_Master_write_slave_address_with_write_req (uint8 address)
- uint8 I2C_Master_write_slave_address_with_read_req (uint8 address)
- uint8 I2C_Master_write_byte (uint8 data)
- uint8 I2C_Master_read_byte (void)

4.16.1 Detailed Description

I2C Module Header File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.16.2 Macro Definition Documentation

4.16.2.1 _XTAL_FREQ

```
#define _XTAL_FREQ 8000000
```

Clock Frequency

4.16.2.2 I2C_BAUDRATE

```
#define I2C_BAUDRATE 9600
```

I2C Baud Rate

4.16.2.3 SCL_PIN

```
#define SCL_PIN 3
```

I2C Clock Pin

4.16.2.4 SDA_PIN

```
#define SDA_PIN 4
```

I2C Data Pin

4.16.3 Function Documentation

4.16.3.1 I2C_Master_Init()

```
void I2C_Master_Init (
             void  )
```

Brief: This is the function to initialize ECU as Master Mode

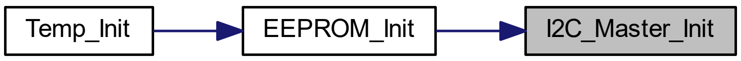
Parameters

void	

Returns

void

Here is the caller graph for this function:



4.16.3.2 I2C_Master_read_byte()

```
uint8 I2C_Master_read_byte (
             void )
```

Brief: This is function to Master read data byte

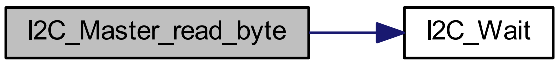
Parameters

void	

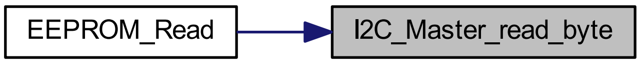
Returns

unsigned char data

Here is the call graph for this function:



Here is the caller graph for this function:



4.16.3.3 I2C_Master_write_byte()

```
uint8 I2C_Master_write_byte (
             uint8 data )
```

Brief: This is function to Master write data byte

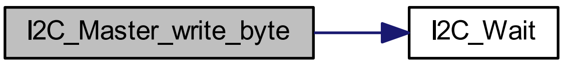
Parameters

data	unsigned char to write it

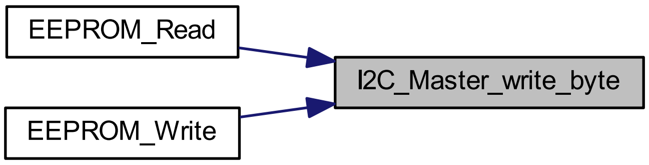
Returns

unsigned char

Here is the call graph for this function:



Here is the caller graph for this function:



4.16.3.4 I2C_Master_write_slave_address_with_read_req()

```
uint8 I2C_Master_write_slave_address_with_read_req (
            uint8 address )
```

Brief: This is function to Master write address byte with read request

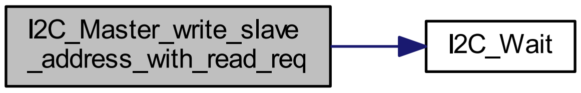
Parameters

address	unsigned char address to select a certain address that you want to read at it

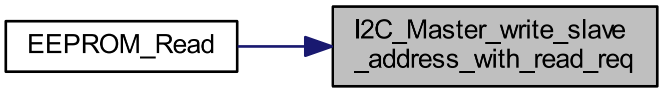
Returns

unsigned char true when finished

Here is the call graph for this function:



Here is the caller graph for this function:



4.16.3.5 I2C_Master_write_slave_address_with_write_req()

```
uint8 I2C_Master_write_slave_address_with_write_req (
            uint8 address )
```

Brief: This is function to Master write address byte with write request

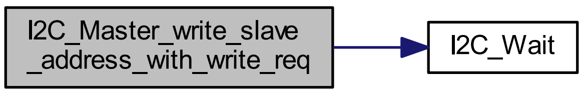
Parameters

address	unsigned char address to select a certain address that you want to write at it

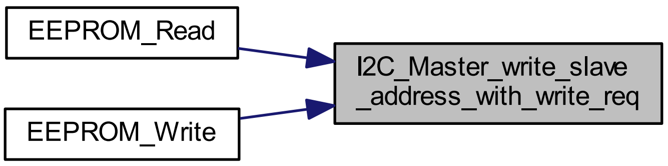
Returns

unsigned char true when finished

Here is the call graph for this function:



Here is the caller graph for this function:



4.16.3.6 I2C_NAck()

```
void I2C_NAck (
            void )
```

Brief: This is the I2C not Ack function

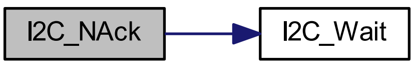
Parameters

void	

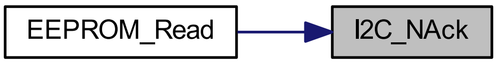
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.16.3.7 I2C_Restart()

```
void I2C_Restart (
            void )
```

Brief: This is the function to Restart I2C communication protocol

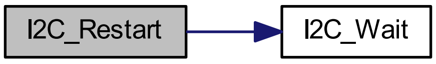
Parameters

void	

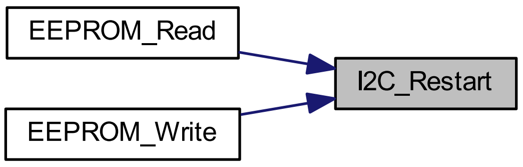
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.16.3.8 I2C_Start()

```
void I2C_Start (
             void  )
```

Brief: This is the function to Start I2C communication protocol

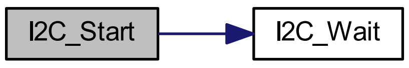
Parameters

void	

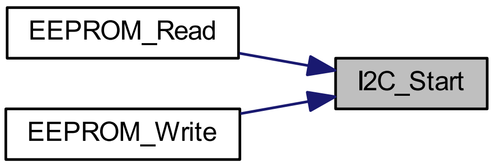
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.16.3.9 I2C_Stop()

```
void I2C_Stop (
             void  )
```

Brief: This is the function to Stop I2C communication protocol

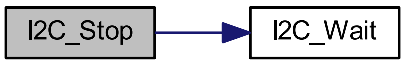
Parameters

void	

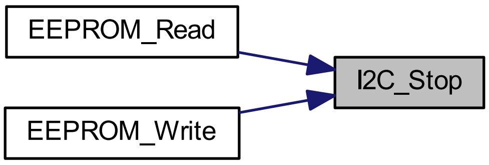
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.16.3.10 I2C_Wait()

```
void I2C_Wait (
             void  )
```

Brief: This is the I2C wait function

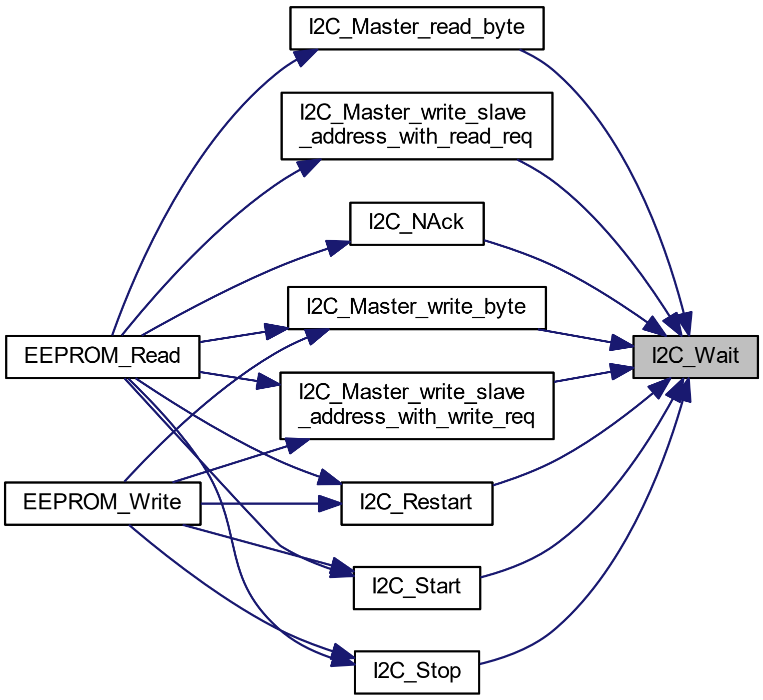
Parameters

void	

Returns

void

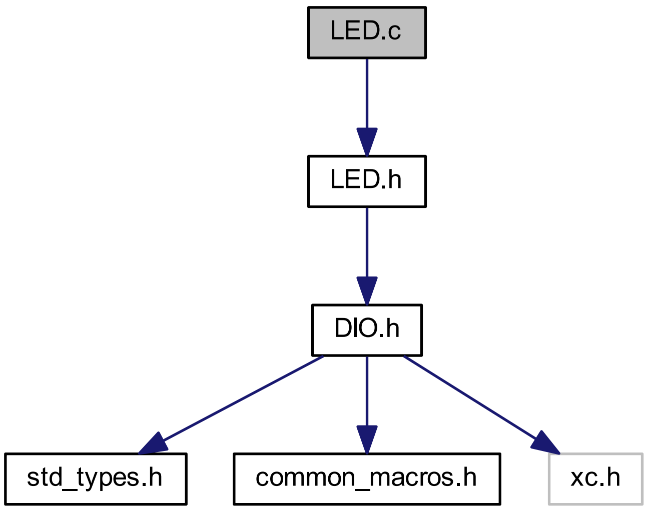
Here is the caller graph for this function:



4.17 LED.c File Reference

LED Module Source File for this program.

```
#include "LED.h"
```
Include dependency graph for LED.c:



Functions

- void LED_OFF (void)
- void LED_Init (void)
- void LED_TOGGLE (void)

4.17.1 Detailed Description

LED Module Source File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.17.2 Function Documentation

4.17.2.1 LED_Init()

```
void LED_Init (
             void  )
```

Brief: This is function to Set LED port direction as OUTPUT

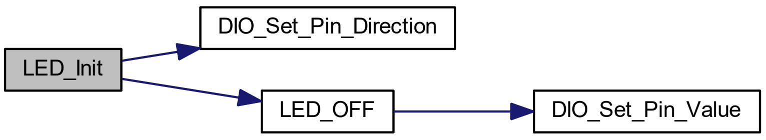
Parameters

void	

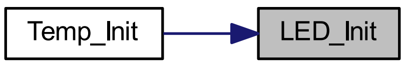
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.17.2.2 LED_OFF()

```
void LED_OFF (
             void  )
```

Brief: This is function to Turn LED OFF

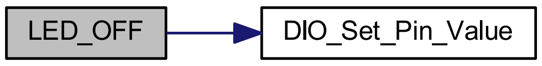
Parameters

void	

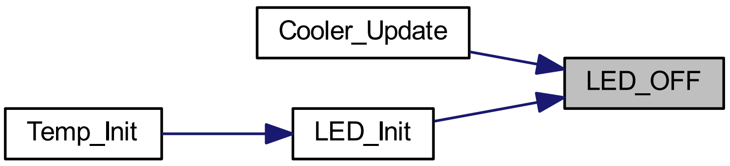
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.17.2.3 LED_TOGGLE()

```
void LED_TOGGLE (
             void  )
```

Brief: This is function to toggle state of LED

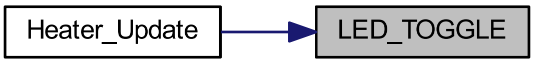
Parameters

void	

Returns

void

Here is the caller graph for this function:

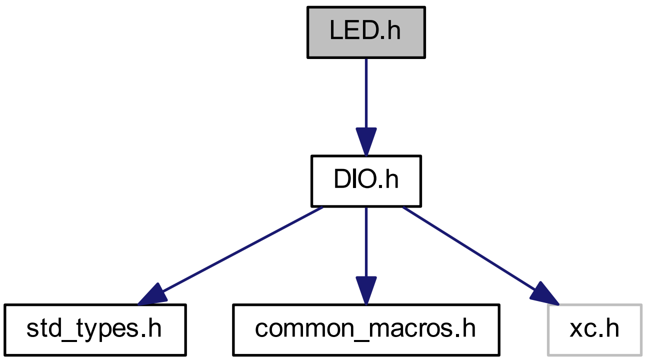


4.18 LED.h File Reference

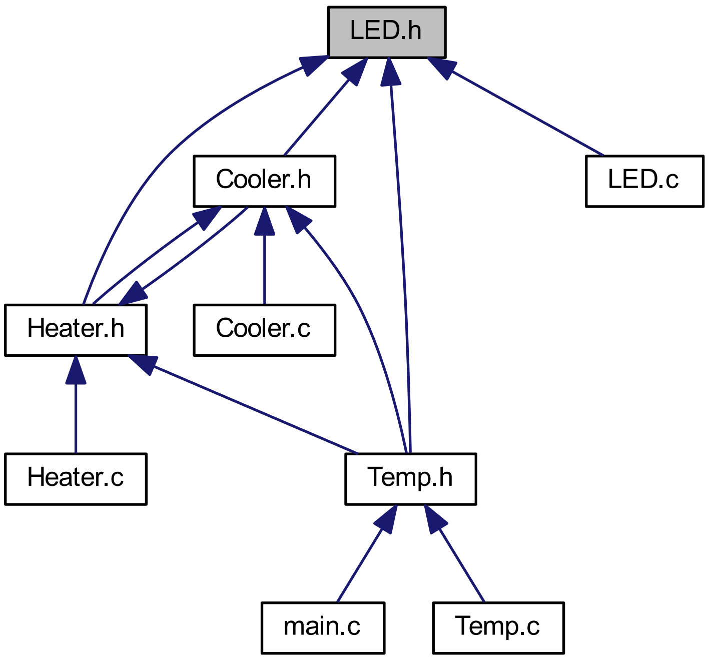
LED Module Header File for this program.

```
#include "DIO.h"
```

Include dependency graph for LED.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define LED_PIN 7
- #define LED_PORT B

Functions

- void LED_OFF (void)
- void LED_Init (void)
- void LED_TOGGLE (void)

4.18.1 Detailed Description

LED Module Header File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.18.2 Macro Definition Documentation

4.18.2.1 LED_PIN

```
#define LED_PIN 7
```

LED Pin

4.18.2.2 LED_PORT

```
#define LED_PORT B
```

LED Port

4.18.3 Function Documentation

4.18.3.1 LED_Init()

```
void LED_Init (
             void  )
```

Brief: This is function to Set LED port direction as OUTPUT

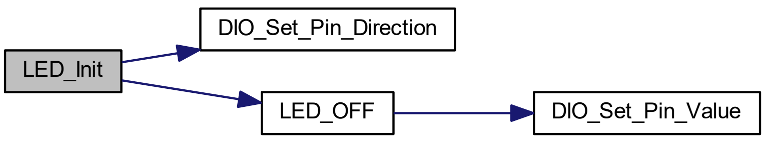
Parameters

void	

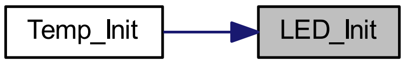
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.18.3.2 LED_OFF()

```
void LED_OFF (
             void  )
```

Brief: This is function to Turn LED OFF

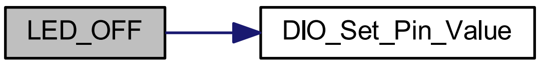
Parameters

void	

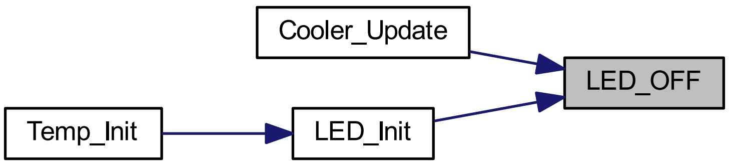
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



4.18.3.3 LED_TOGGLE()

```
void LED_TOGGLE (
             void  )
```

Brief: This is function to toggle state of LED

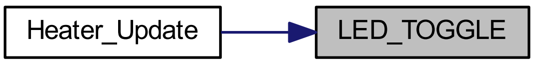
Parameters

void	

Returns

void

Here is the caller graph for this function:



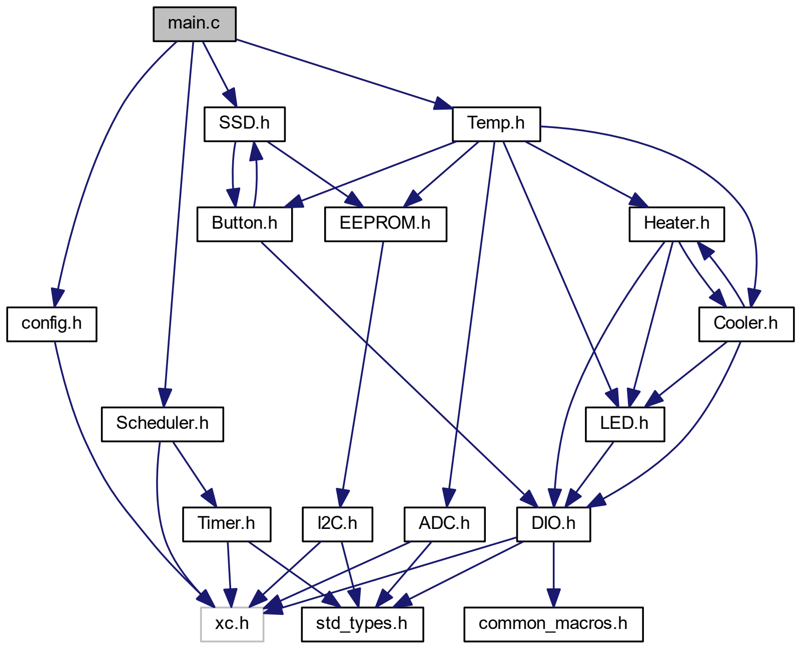
4.19 main.c File Reference

main Source File for this program

```
#include "config.h"
#include "SSD.h"
```

```
#include "Scheduler.h"
#include "Temp.h"
```

Include dependency graph for main.c:



Macros

- #define **_XTAL_FREQ** 8000000

Functions

- void **main** (void)

4.19.1 Detailed Description

main Source File for this program

Author

Mohammed Awwad

Date

10/7/2020

Version

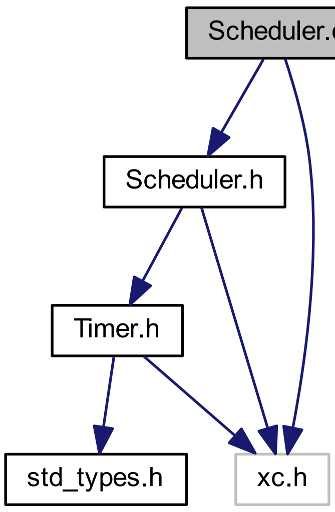
1.0

4.20 Scheduler.c File Reference

Scheduler Source File for this program.

```
#include "Scheduler.h"
#include <xc.h>
```

Include dependency graph for Scheduler.c:



Functions

- void SCH_Init_T1 (void)
- void SCH_Update (void)
- void **__interrupt** () ISR(void)
- uint8 **SCH_Add_Task** (void(∗pFunction)(), const uint16 DELAY, const uint16 PERIOD)
- void SCH_Dispatch_Tasks (void)
- void SCH_Start (void)
- void SCH_Go_To_Sleep (void)

Variables

- sTask_t **SCH_tasks_G** [SCH_MAX_TASKS]

4.20.1 Detailed Description

Scheduler Source File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.20.2 Function Documentation

4.20.2.1 SCH_Dispatch_Tasks()

```
void SCH_Dispatch_Tasks (
             void   )
```

Brief: This is the 'dispatcher' function. When a task (function) is due to run, SCH_Dispatch_Tasks() will run it. This function must be called (repeatedly) from the main loop.

Parameters

void	

Returns

void

4.20.2.2 SCH_Go_To_Sleep()

```
void SCH_Go_To_Sleep (
             void   )
```

Brief: This is the Go to Sleep function. This scheduler enters 'idle mode' between clock ticks to save power. The next clock tick will return the processor to the normal operating state. Note: a slight performance improvement is possible if this function is implemented as a macro, or if the code here is simply pasted into the 'dispatch' function. $***$ ADAPT AS REQUIRED FOR YOUR HARDWARE $***$

Parameters

void	

Returns

void

4.20.2.3 SCH_Init_T1()

```
void SCH_Init_T1 (
             void  )
```

Brief: This is the Scheduler initialization function. Prepares scheduler data structures and sets up timer interrupts at required rate. You must call this function before using the scheduler.

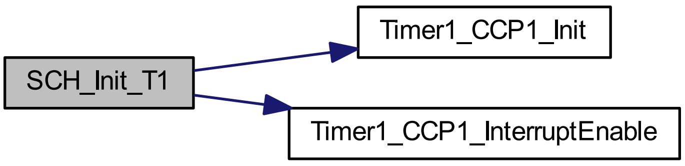
Parameters

void	

Returns

void

Here is the call graph for this function:



4.20.2.4 SCH_Start()

```
void SCH_Start (
             void  )
```

Brief: This is the Scheduler start function. Starts the scheduler, by enabling interrupts. NOTE: Usually called after all regular tasks are added,to keep the tasks synchronized. NOTE: ONLY THE SCHEDULER INTERRUPT SHOULD BE ENABLED!!!

Parameters

void	

Returns

void

4.20.2.5 SCH_Update()

```
void SCH_Update (
             void  )
```

Brief: This is the the scheduler ISR. It is called at a rate determined by the timer settings in SCH_Init_T1(). This version is triggered by Timer 1 interrupts:

Parameters

void	

Returns

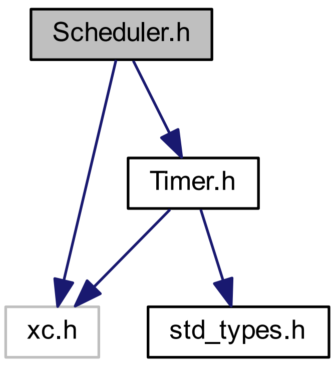
void

4.21 Scheduler.h File Reference

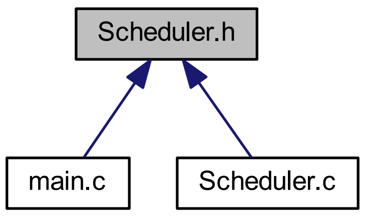
Scheduler Header File for this program.

```
#include <xc.h>
#include "Timer.h"
```

Include dependency graph for Scheduler.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sTask_t

Macros

- #define SCH_MAX_TASKS (4)

Functions

- void SCH_Init_T1 (void)
- void SCH_Update (void)
- uint8 SCH_Add_Task (void(∗pFunction)(void), const uint16, const uint16)
- void SCH_Dispatch_Tasks (void)
- void SCH_Start (void)
- void SCH_Go_To_Sleep (void)

4.21.1 Detailed Description

Scheduler Header File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.21.2 Macro Definition Documentation

4.21.2.1 SCH_MAX_TASKS

```
#define SCH_MAX_TASKS (4)
```

The maximum number of tasks required at any one time during the execution of the program

4.21.3 Function Documentation

4.21.3.1 SCH_Add_Task()

```
uint8 SCH_Add_Task (
             void(*)(void) pFunction,
             const uint16,
             const uint16 )
```

Brief: This is the the SCH_Add_Task Causes a task (function) to be executed at regular intervals or after a user-defined delay

Parameters

pointer	to function - The name of the function which is to be scheduled. NOTE: All scheduled functions must be 'void, void' - that is, they must take no parameters, and have a void return type.
DELAY	- The interval (TICKS) before the task is first executed
PERIOD	- If 'PERIOD' is 0, the function is only called once, at the time determined by 'DELAY'. If PERIOD is non-zero,then the function is called repeatedly at an interval determined by the value of PERIOD

Returns

Returns the position in the task array at which the task has been added. If the return value is SCH_MAX_↩ TASKS then the task could not be added to the array (there was insufficient space). If the return value is $<$ SCH_MAX_TASKS, then the task was added successfully.

4.21.3.2 SCH_Dispatch_Tasks()

```
void SCH_Dispatch_Tasks (
             void )
```

Brief: This is the 'dispatcher' function. When a task (function) is due to run, SCH_Dispatch_Tasks() will run it. This function must be called (repeatedly) from the main loop.

Parameters

void	

Returns

void

4.21.3.3 SCH_Go_To_Sleep()

```
void SCH_Go_To_Sleep (
            void  )
```

Brief: This is the Go to Sleep function. This scheduler enters 'idle mode' between clock ticks to save power. The next clock tick will return the processor to the normal operating state. Note: a slight performance improvement is possible if this function is implemented as a macro, or if the code here is simply pasted into the 'dispatch' function. ∗∗∗ ADAPT AS REQUIRED FOR YOUR HARDWARE ∗∗∗

Parameters

void	

Returns

void

4.21.3.4 SCH_Init_T1()

```
void SCH_Init_T1 (
            void  )
```

Brief: This is the Scheduler initialization function. Prepares scheduler data structures and sets up timer interrupts at required rate. You must call this function before using the scheduler.

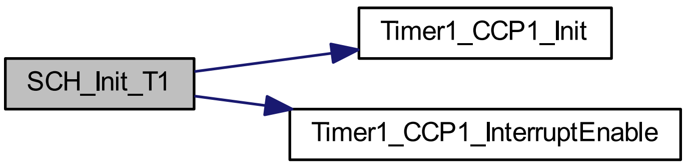
Parameters

void	

Returns

void

Here is the call graph for this function:



4.21.3.5 SCH_Start()

```
void SCH_Start (
            void  )
```

Brief: This is the Scheduler start function. Starts the scheduler, by enabling interrupts. NOTE: Usually called after all regular tasks are added,to keep the tasks synchronized. NOTE: ONLY THE SCHEDULER INTERRUPT SHOULD BE ENABLED!!!

Parameters

void	

Returns

void

4.21.3.6 SCH_Update()

```
void SCH_Update (
            void  )
```

Brief: This is the the scheduler ISR. It is called at a rate determined by the timer settings in SCH_Init_T1(). This version is triggered by Timer 1 interrupts:

Parameters

void	

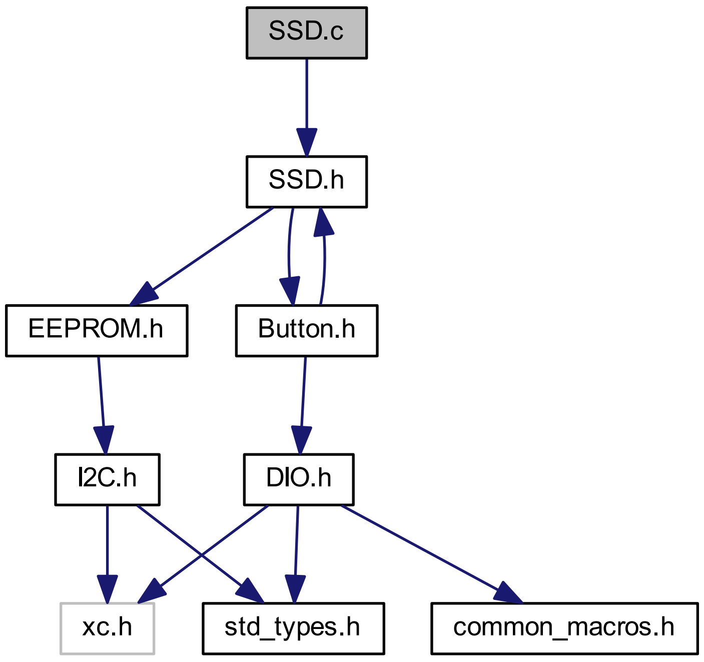
Returns

void

4.22 SSD.c File Reference

7-Segment Display Source File for this program

```
#include "SSD.h"
```
Include dependency graph for SSD.c:



Functions

- uint8 display7s (uint8 number)
- void SSD_Init (void)
- void **SSD_Display_OFF** (void)
- void **SSD_Display_Current_Temp** (void)
- void **SSD_Display_Set_Point_Temp** (void)
- void SSD_Flash (void)
- void SSD_Update (void)

Variables

- System_States_t **System_State**
- uint16 **Temp**
- uint16 **Set_Point_Temp**
- uint8 **SW_UP_isPressed**
- uint8 **SW_DOWN_isPressed**
- uint8 **digit0** =1
- uint8 **counter** =0
- uint8 **Temp_Updated** =0
- uint8 **KeepMeHere_flag** =0
- Flash_State_t **Flash_State** = SSD_OFF
- SSD_MODE_States_t **SSD_MODE_State** = SSD_NORMAL

4.22.1 Detailed Description

7-Segment Display Source File for this program

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.22.2 Function Documentation

4.22.2.1 display7s()

```
uint8 display7s (
            uint8 nubmer )
```

Brief: This is the Function to display number on 7 segment

Parameters

nubmer	unsigned char

Returns

unsigned char

4.22.2.2 SSD_Flash()

```
void SSD_Flash (
             void  )
```

Brief: This is the Function to blink set point temperature every 1 sec when SSD at SSD setting mode

Parameters

void	

Returns

void

4.22.2.3 SSD_Init()

```
void SSD_Init (
             void  )
```

Brief: This is the SSD initialization function to initialize the direction of SSD port

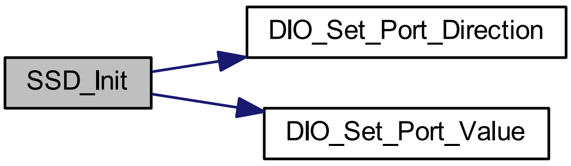
Parameters

void	

Returns

void

Here is the call graph for this function:



4.22.2.4 SSD_Update()

```
void SSD_Update (
            void  )
```

Brief: This is the SSD Update Task to update the SSD Mode every 25 ms

Parameters

void	

Returns

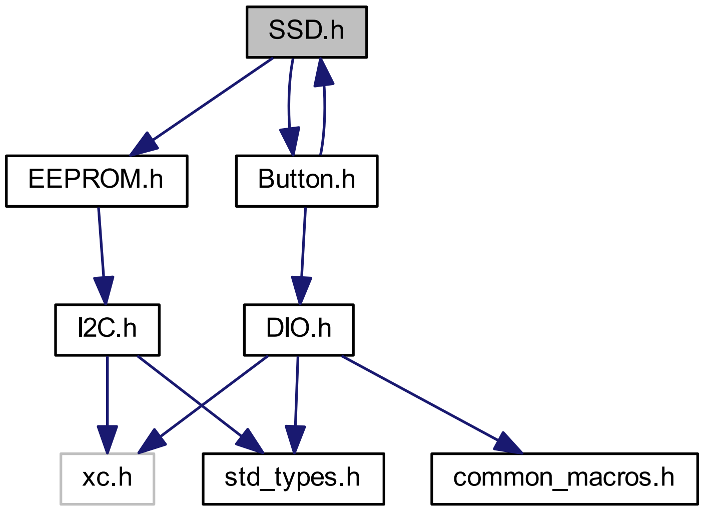
void

4.23 SSD.h File Reference

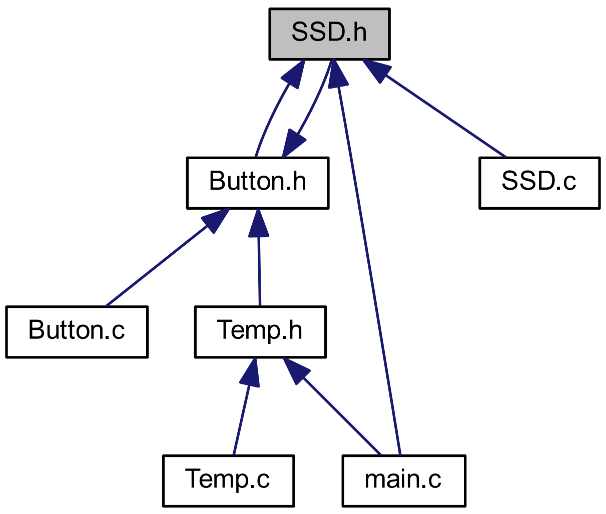
7-Segment Display Header File for this program

```
#include "EEPROM.h"
#include "Button.h"
```

Include dependency graph for SSD.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define SSD_CTR_PORT A
- #define SSD_DTA_PORT D
- #define DIGIT_1 0x10
- #define DIGIT_10 0x20
- #define MAX_TEMP 75
- #define MIN_TEMP 35

Enumerations

- enum **Flash_State_t** { **SSD_OFF**, **SSD_ON** }
- enum **SSD_MODE_States_t** { **SSD_NORMAL**, **SSD_SETTING** }

Functions

- uint8 display7s (uint8 nubmer)
- void SSD_Init (void)
- void SSD_Update (void)
- void SSD_Display_Temp (void)
- void SSD_Flash (void)
- void SSD_Display_Set_Point (void)

4.23.1 Detailed Description

7-Segment Display Header File for this program

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.23.2 Macro Definition Documentation

4.23.2.1 DIGIT_1

```
#define DIGIT_1 0x10
```

SSD Enable 1 digit

4.23.2.2 DIGIT_10

```
#define DIGIT_10 0x20
```

SSD Enable 10 digit

4.23.2.3 MAX_TEMP

```
#define MAX_TEMP 75
```

Maximum temperature can be reach

4.23.2.4 MIN_TEMP

```
#define MIN_TEMP 35
```

Minimum temperature can be reach

4.23.2.5 SSD_CTR_PORT

```
#define SSD_CTR_PORT A
```

SSD Control Port

4.23.2.6 SSD_DTA_PORT

```
#define SSD_DTA_PORT D
```

SSD Data Port

4.23.3 Function Documentation

4.23.3.1 display7s()

```
uint8 display7s (
            uint8 nubmer )
```

Brief: This is the Function to display number on 7 segment

Parameters

nubmer	unsigned char

Returns

unsigned char

4.23.3.2 SSD_Display_Set_Point()

```
void SSD_Display_Set_Point (
            void )
```

Brief: This is the Function to display set point temperature when current water temperature at set point interval

Parameters

void	

Returns

void

4.23.3.3 SSD_Display_Temp()

```
void SSD_Display_Temp (
            void  )
```

Brief: This is the Function to blink set point temperature every 1 sec when SSD at SSD setting mode

Parameters

void	

Returns

void

4.23.3.4 SSD_Flash()

```
void SSD_Flash (
            void  )
```

Brief: This is the Function to blink set point temperature every 1 sec when SSD at SSD setting mode

Parameters

void	

Returns

void

4.23.3.5 SSD_Init()

```
void SSD_Init (
            void  )
```

Brief: This is the SSD initialization function to initialize the direction of SSD port

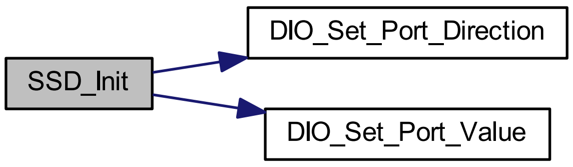
Parameters

void	

Returns

void

Here is the call graph for this function:



4.23.3.6 SSD_Update()

```
void SSD_Update (
            void  )
```

Brief: This is the SSD Update Task to update the SSD Mode every 25 ms

Parameters

void	

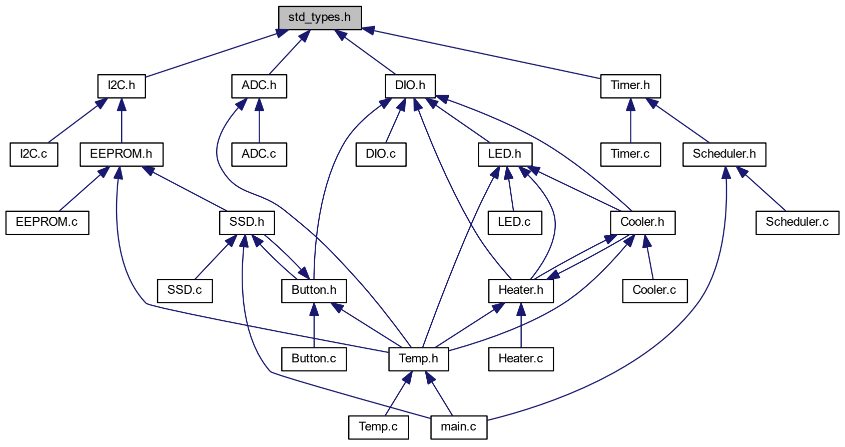
Returns

void

4.24 std_types.h File Reference

Standard Types Header File for this program.

This graph shows which files directly or indirectly include this file:



Macros

- #define **NULLPTR** ((void∗)0)

Typedefs

- typedef unsigned char **uint8**
- typedef signed char **sint8**
- typedef unsigned short **uint16**
- typedef signed short **sint16**
- typedef unsigned long **uint32**
- typedef signed long **sint32**

4.24.1 Detailed Description

Standard Types Header File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

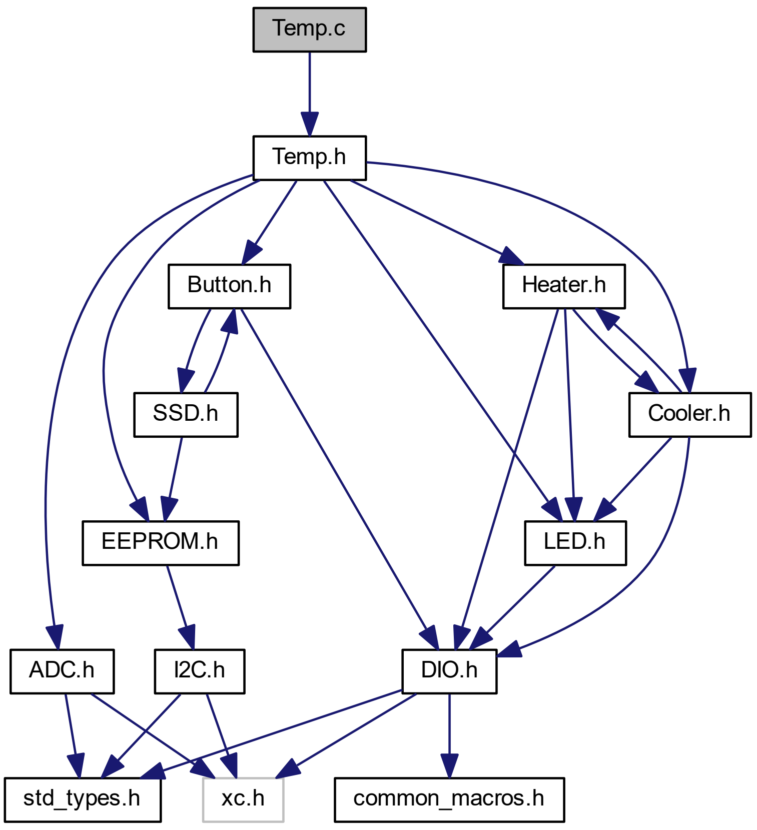
1.0

4.25 Temp.c File Reference

Temperature Source File for this program.

```
#include "Temp.h"
```

Include dependency graph for Temp.c:



Functions

- void Temp_Init (void)
- void Temp_Get (void)
- void Temp_Update (void)

Variables

- System_States_t **System_State**
- uint8 **Temp_Updated**
- uint16 **Temp**
- uint8 **Flag** =0
- uint16 **Set_Point_Temp** =60
- uint8 **Last_Set_Point_Temp**
- Temp_States_t **Temp_State** = Temp_Min_State

4.25.1 Detailed Description

Temperature Source File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.25.2 Function Documentation

4.25.2.1 Temp_Get()

```
void Temp_Get (
            void )
```

Brief: This is the get Temperature Task to get sample of temperature every 100ms and calculate the average to update temperature value in Temp_Update Function

Parameters

void	

Returns

void

4.25.2.2 Temp_Init()

```
void Temp_Init (
            void )
```

Brief: This is the Temperature initialization function to initialize ADC , Heater , Cooler , EEPROM then check if the system has run never run run before initialize set point temp by 60

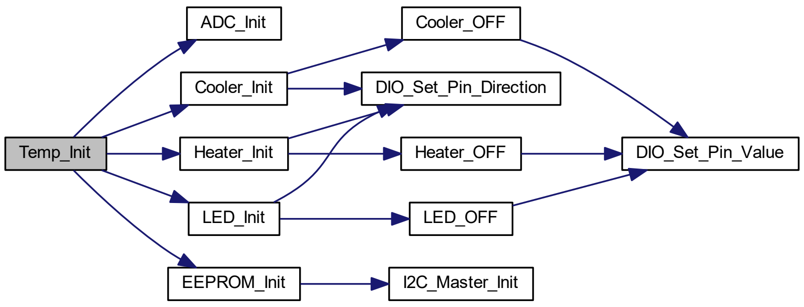
Parameters

void	

Returns

void

Here is the call graph for this function:



4.25.2.3 Temp_Update()

```
void Temp_Update (
            void )
```

Brief: This is the Temperature Update Task to update temperature value every 1000ms by using cooler and heater elements and check every change in temp at setting mode write it at certain address at EEPROM

Parameters

void	

Returns

void

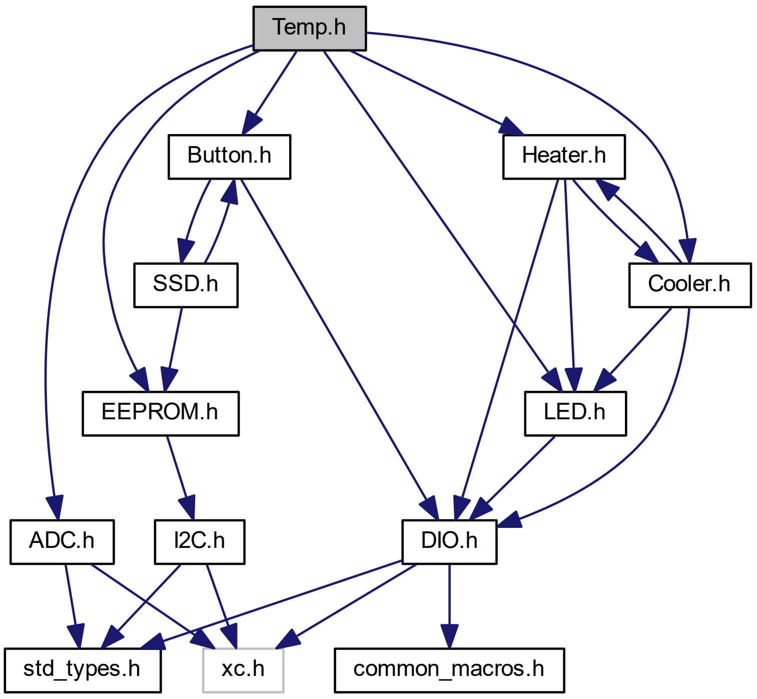
4.26 Temp.h File Reference

Temperature Header File for this program.

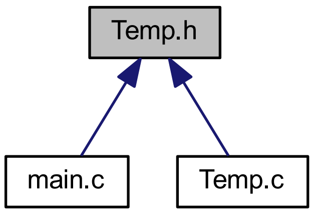
```
#include "ADC.h"
#include "EEPROM.h"
```

```
#include "Button.h"
#include "LED.h"
#include "Heater.h"
#include "Cooler.h"
```

Include dependency graph for Temp.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define NOT_WRITTEN_BEFOR (0xFF)

Enumerations

- enum **Temp_States_t** { **Temp_Min_State**, **Temp_Set_Point_State**, **Temp_Max_State** }

Functions

- void Temp_Init (void)
- void Temp_Update (void)
- void Temp_Get (void)

4.26.1 Detailed Description

Temperature Header File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.26.2 Macro Definition Documentation

4.26.2.1 NOT_WRITTEN_BEFOR

```
#define NOT_WRITTEN_BEFOR (0xFF)
```

this constant used to check if there is stord ste point temperature

4.26.3 Function Documentation

4.26.3.1 Temp_Get()

```
void Temp_Get (
             void  )
```

Brief: This is the get Temperature Task to get sample of temperature every 100ms and calculate the average to update temperature value in Temp_Update Function

Parameters

void	

Returns

void

4.26.3.2 Temp_Init()

```
void Temp_Init (
            void )
```

Brief: This is the Temperature initialization function to initialize ADC , Heater , Cooler , EEPROM then check if the system has run never run run before initialize set point temp by 60

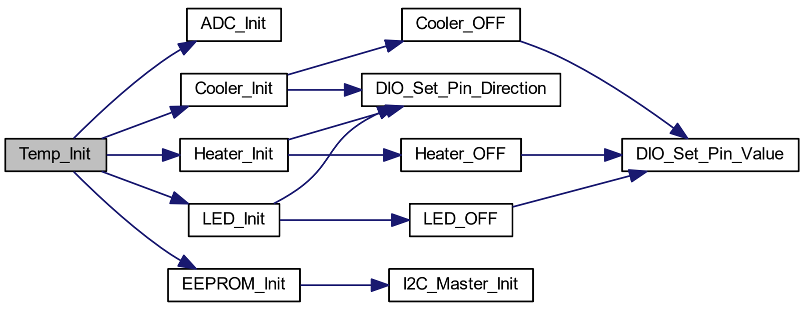
Parameters

void	

Returns

void

Here is the call graph for this function:



4.26.3.3 Temp_Update()

```
void Temp_Update (
            void )
```

Brief: This is the Temperature Update Task to update temperature value every 1000ms by using cooler and heater elements and check every change in temp at setting mode write it at certain address at EEPROM

Parameters

void	

Returns

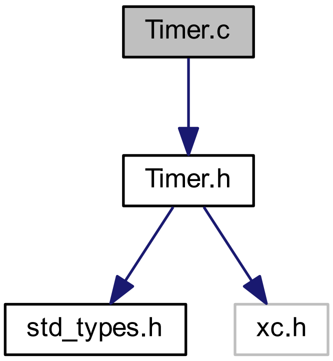
void

4.27 Timer.c File Reference

Timer Module Source File for this program.

```
#include "Timer.h"
```

Include dependency graph for Timer.c:



Functions

- void Timer1_CCP1_Init ()
- void Timer1_CCP1_InterruptEnable ()

4.27.1 Detailed Description

Timer Module Source File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.27.2 Function Documentation

4.27.2.1 Timer1_CCP1_Init()

```
void Timer1_CCP1_Init (
            void )
```

Brief: This is Timer 1 initialization at CCP Mode

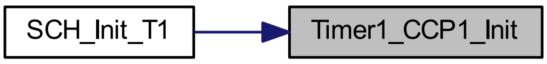
Parameters

void	

Returns

void

Here is the caller graph for this function:



4.27.2.2 Timer1_CCP1_InterruptEnable()

```
void Timer1_CCP1_InterruptEnable (
            void )
```

Brief: This is Timer 1 Interrupt enable at CCP Mode

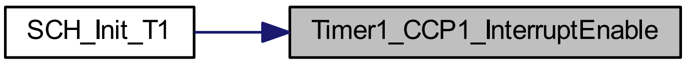
Parameters

void	

Returns

void

Here is the caller graph for this function:

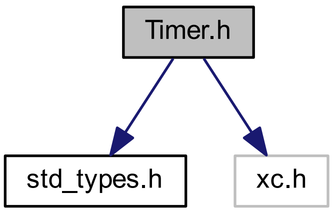


4.28 Timer.h File Reference

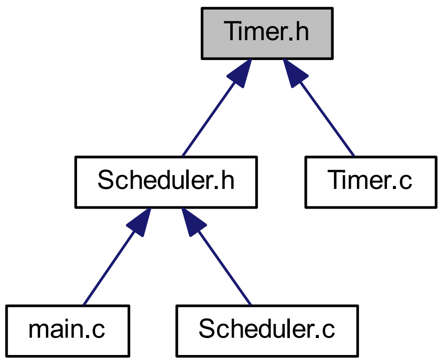
Timer Module Header File for this program.

```
#include "std_types.h"
#include <xc.h>
```

Include dependency graph for Timer.h:



This graph shows which files directly or indirectly include this file:



Functions

- void Timer1_CCP1_Init (void)
- void Timer1_CCP1_InterruptEnable (void)

4.28.1 Detailed Description

Timer Module Header File for this program.

Author

Mohammed Awwad

Date

10/7/2020

Version

1.0

4.28.2 Function Documentation

4.28.2.1 Timer1_CCP1_Init()

```
void Timer1_CCP1_Init (
            void  )
```

Brief: This is Timer 1 initialization at CCP Mode

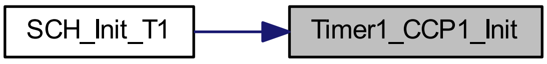
Parameters

void	

Returns

void

Here is the caller graph for this function:



4.28.2.2 Timer1_CCP1_InterruptEnable()

```
void Timer1_CCP1_InterruptEnable (
            void  )
```

Brief: This is Timer 1 Interrupt enable at CCP Mode

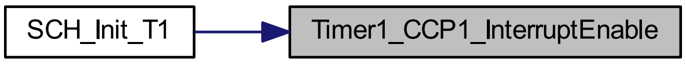
Parameters

void	

Returns

void

Here is the caller graph for this function:



Index